

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: «Дослідження інформаційних технологій керування корпоративними даними та розробка проєкту системи аналізу ефективності сховища даних ІТ-компанії»

Виконав: студент групи 6145 м

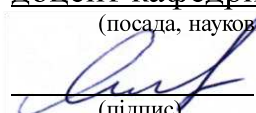


Поліщук В.В.

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,

(посада, науковий ступінь вчене звання)



Михелєв І.Л.

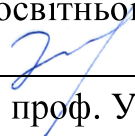
м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 124 "Системний аналіз"
Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


д.ф-м.н, проф. Ушкац М.В.

" ____ " _____ 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту Поліщуку Володимир Володимировичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження інформаційних технологій керування корпоративними даними та розробка проєкту системи аналізу ефективності сховища даних ІТ-компанії

Керівник роботи к.т.н., доцент Михелєв Ігор Леонідович

Затверджені наказом ректора № 1055-уч. від "08" жовтня 2024 року.

2. Термін подання роботи: 15 грудня 2024 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів)

5. Перелік презентаційних матеріалів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Михелєв І.Л.	05.09.2023	22.12.2023
2	Михелєв І.Л.	23.12.2023	15.06.2024
3	Михелєв І.Л.	15.06.2024	25.09.2024
4	Михелєв І.Л.	25.09.2024	14.11.2024
5	Михелєв І.Л.	25.09.2024	14.11.2024
6	Михелєв І.Л.	25.09.2024	14.11.2024

7. Дата видачі завдання 05 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2024	
3	Розділ з розробки проектних рішень по системі (та її частинах)	15.06.2024	
4	Реалізація проекту та розробка робочої документації	25.09.2024	
5	Обчислення економічної ефективності	14.11.2024	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2024	
7	Оформлення пояснювальної записки	14.11.2024	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2024	
9	Подання магістерської роботи рецензенту	20.11.2024	
10	Подання магістерської роботи на державний захист	15.12.2024	

Студент


(підпис)

Керівник роботи

Поліщук В.В.
(ПІБ)

Михелєв І.Л.
(ПІБ)

АНОТАЦІЯ

Робота “Дослідження інформаційних технологій керування корпоративними даними та розробка проєкту системи аналізу ефективності сховища даних ІТ-компанії” присвячена дослідженню технологій організації ефективних корпоративних сховищ даних автоматизованих інформаційних систем ІТ-компаній та механізмів аналізу ефективності інформаційного сховища на основі визначених критеріїв оцінки ефективності.

В роботі представлені результати аналізу технологій збереження, організації структури та доступу до бізнес-даних, розглянуті та визначені особливості застосування інструментальних засобів бізнес-аналізу, визначені критерії оцінки ефективності корпоративних сховищ даних автоматизованих інформаційних систем.

В роботі запропонований науково-обґрунтований підхід до оцінки ефективності корпоративних сховищ даних та аналізу ефективності сховищ даних, наведені результати застосування розроблених моделей та механізмів аналізу ефективності у реальних проєктах.

Ключові слова: системний аналіз, інформаційні технології, корпоративні данні, сховища даних.

ABSTRACT

The work “Advancement of information technologies for the management of corporate data and development of a system design for analyzing the effectiveness of the data collection of an IT company” is dedicated to advanced technologies for organizing effective corporate data warehouses from automated information systems of IT companies and mechanisms for analyzing the effectiveness of the information warehouse based on the relevant criteria for assessing the effectiveness.

The work presents the results of the analysis of saving technologies, the organization of structure and access to business data, the specific features of establishing instrumental methods for business analysis, and the identification of evaluation criteria. the effectiveness of corporate data collections from automated information systems.

In the process of scientific research, we approach the assessment of the effectiveness of corporate data collections and the analysis of the effectiveness of data collections, the results of the establishment of fragmented models and mechanisms for analyzing the effectiveness of data. real projects.

Keywords: system analysis, information technologies, corporate data, data warehouses.

ЗМІСТ

Вступ

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ В КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд методів і засобів збереження та аналізу даних в корпоративних інформаційних системах

1.1.1. Моделювання бізнес процесів в корпоративних інформаційних системах

1.1.2. Реляційні СКБД

1.1.3. NoSQL СКБД

1.1.4. ОО СКБД

1.1.5. Документно-орієнтовані бази даних

1.1.6. Розподілені бази даних

1.1.7. Висновки щодо методів та засобів збереження даних

1.2 Огляд інтернет-сервісів та протоколів доступу до даних корпоративних інформаційних систем

1.2.1. Механізми доступу та сервіси

1.2.2. Інтернет-сервери HTTP та HTTPS

1.2.3 Особливості використання файлових сховищ в задачах бізнес-аналітики

1.2.4 Задачі бізнес-аналітики і технологія OLAP

1.2.5. Висновки щодо методів та засобів передачі даних

1.3. Проблеми доступу до даних і аналізу даних в проектах корпоративних інформаційних систем

1.3.1. Проблеми захисту даних та каналів передачі даних

1.3.2. Проблеми, викликані модифікацією структур даних

1.3.3. Проблеми синхронізації даних в корпоративних сховищах даних

1.3.4. Проблеми, що визначаються предметною областю проектів

1.3.5. Проблеми, що визначаються інструментальним забезпеченням

1.3.6. Проблеми обсягу даних корпоративних інформаційних систем

1.4. Постановка задачі дослідження

1.5. Висновки до розділу 1

РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ ТА МЕХАНІЗМІВ ОЦІНКИ

ЕФЕКТИВНОСТІ СХОВИЩ ДАНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1. Визначення критеріїв оцінки методів та засобів розробки проектів сховищ даних корпоративних інформаційних систем

2.1.1. Визначення критеріїв оцінки методів та засобів збереження і аналізу даних в проектах корпоративних інформаційних систем

2.1.2. Визначення критеріїв оцінки методів та засобів розробки проектів активних сховищ даних корпоративних інформаційних систем

2.2. Розробка моделей оцінки якості механізмів збереження та доступу до активних сховищ даних

2.2.1. Критерії оцінки якості технологій збереження та доступу до активних сховищ даних корпоративних інформаційних систем

2.2.2. Механізми забезпечення якості активних сховищ даних корпоративних інформаційних систем

2.2.3. Механізми забезпечення якості доступу до активних сховищ даних корпоративних інформаційних систем

2.2.4. Механізми доступу та збереження даних клієнта активних сховищ даних корпоративних інформаційних систем

2.3. Розробка моделей вибору методів та засобів збереження та доступу і аналізу даних в проектах корпоративних інформаційних систем

2.4. Вибір інструментального забезпечення розробки проектів аналізу ефективності сховищ даних корпоративних інформаційних систем

2.5. Визначення властивостей мов веб-програмування корпоративних інформаційних систем з веб-доступом

2.6. Висновки до розділу 2

РОЗДІЛ 3. ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ ОЦІНКИ ЕФЕКТИВНОСТІ СХОВИЩ ДАНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

3.1. Розробка адаптивної моделі навантаженого сервера активного сховища даних корпоративної інформаційної системи

3.2. Результати тестування активного сховища даних для різних “engines” РСКБД MySQL

3.3. Висновки до розділу 3

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

Додаток А. Технічне завдання на розробку програмного забезпечення підсистеми аналізу ефективності сховища даних

Додаток Б. Приклад реалізації моделей даних з механізмами підсистеми аналізу ефективності сховища даних інформаційної системи (фрагмент)

ВСТУП

Корпоративні клієнто-орієнтовані технології останнім часом зазнали стрімкого розвитку — на їх основі надаються послуги, навчальний контент, реклама тощо. Обсяги даних, що в той чи інший спосіб використовуються для реалізації таких технологій в окремо взятому бізнесі, становлять сотні терабайт і продовжують зростати. Останнім часом також широко розвивається напрямок корпоративних сховищ даних, а системи управління (або керування) базами даних (СКБД, СКБД) є одним з найбільш поширених класів прикладних систем для корпоративних серверів даних. Це пов'язано з тим, що обсяги даних, що в той чи інший спосіб доступні в корпоративному секторі, в тому числі і в Інтернет, вже становлять сотні терабайт і продовжують зростати, а спеціальні алгоритми традиційних СКБД не можуть забезпечити прийнятний час відгуку на запитах даних в задачах аналізу та побудови корпоративної звітності. Саме тому для вирішення таких задач останнім часом використовують спеціалізовані рішення для аналізу і обробки даних на рівні систем збереження даних, які, у свою чергу, виконуються на спеціальних обчислювальних системах з масивно-паралельною архітектурою, що дозволяють здійснити такий аналіз даних на значних обсягах даних. Такі архітектури дозволяють розпаралелити обробку даних і винести частину обробки з рівня СКБД на рівень серверів зберігання.

Часткове вирішення таких задач також забезпечують хмарні обчислення (Cloud computing). Ця технологія дуже приваблива для користувачів, оскільки дозволяє відносно просто отримати через Інтернет у тимчасове користування деякий сервіс для зберігання і аналізу даних. Наприклад, можна замовити комп'ютер із заданим об'ємом пам'яті, дисків, кількістю процесорів, з попередньо встановленими ОС і СКБД або іншими пакетами ПЗ і працювати з ним зі свого офісу, хоча фізично і обладнання, і ПО розташовуються і обслуговуються в віддаленому чужому центрі обслуговування даних. В інтернет поширені служби, де можна замовити потрібний дисковий ресурс і зберігати на ньому свої файли. Прикладами Cloud Computing є сервіси зберігання (Amazon

S3), обчислювальні сервіси (Google App Engine, Amazon EC2) і сервіси даних (Amazon SimpleDB, Microsoft SQL Server Data Services, Google's Datastore).

Зважаючи на вимоги Інтернет-сервісів до обробки значних обсягів даних та активного подання запитів на доступ до даних сучасні СКБД повинні відповідати цим вимогам. Крім того, Інтернет-доступ до СКБД, в тому числі в середовищі cloud computing, висуває підвищені вимоги до СКБД: з точки зору захисту і розмежування доступу (дані розміщуються на чужих комп'ютерах); масштабованості (коли користувач сервісу може розширити вимоги до обсягів дискового простору, потужності та числа процесорів). Оскільки управління такими системами відбувається через Інтернет, це також визначає вимоги до методів та механізмів вибору технологій зберігання даних, вдосконалення засобів адміністрування баз даних та реорганізації їх структури відповідно до бізнес-процесів інтернет-користувачів СКБД.

Таким чином істотною проблемою розробки та експлуатації (у тому числі – розширення) корпоративних інформаційних систем є обмеження, пов'язані з механізмами доступу і аналізу до даних в мережі Інтернет в частині вибірки, передачі, обробки та захисту даних; обмеження, пов'язані зі складністю моделювання бізнес процесів, що постійно змінюються та вимагають модифікації алгоритмів обробки корпоративних даних та самих структур даних. Як наслідок, розширення функціональності таких систем часто супроводжується змінами в структурах даних корпоративної бази даних, які складаються в результаті розвитку таких систем, що для корпоративних систем є нетривіальним завданням і часто вимагає суттєвої переробки модулів, інтерфейсів і архітектури систем, які використовують такі дані і виконують їх аналіз.

Зважаючи на значну кількість технологій організації, збереження і аналізу даних в системах баз даних, механізмів доступу до даних та можливостей їх застосування в корпоративних системах та, зокрема, проєкт сховища даних корпоративної інформаційної системах, постає задача науково-обґрунтованого

вибору механізмів збереження даних в сховищах даних та механізмів доступу і аналізу до цих даних.

Актуальність роботи визначається широким застосуванням корпоративних автоматизованих інформаційних систем в сферах діяльності, де необхідне використання персональних даних персоналу і клієнтів, та наявними проблемами, викликаними необхідністю максимально обмежити використання таких даних для уникнення їх компрометації, витоку або втрати.

Метою дослідження є підвищення ефективності процесів прийняття рішень щодо аналізу стану та оцінки ефективності корпоративного сховища даних в клієнто-орієнтованих компаніях на основі науково обґрунтованих моделей та механізмів.

Об'єктом дослідження є технології керування сховищами даних АІС клієнто-орієнтованих ІТ-компаній, предметом дослідження є процеси аналізу та оцінки ефективності корпоративного сховища даних АІС в клієнто-орієнтованих компаніях.

Для досягнення мети дослідження визначені наступні задачі дослідження: виконати аналіз наявних технологій збереження даних АІС клієнто-орієнтованих компаній, визначити критерії оцінки ефективності використання сховищ даних, розробити моделі та механізми аналізу і оцінки ефективності сховищ даних АІС клієнто-орієнтованих ІТ-компаній.

Наукова новизна дослідження полягає у розробці моделей та механізмів аналізу та оцінки ефективності сховищ даних АІС клієнто-орієнтованих ІТ-компаній.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ В КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд методів і засобів збереження та аналізу даних в корпоративних інформаційних системах

1.1.1. Моделювання бізнес процесів в корпоративних інформаційних системах

Вирішення задач моделювання бізнес процесів компанії, визначення відповідності організації структур даних цим бізнес-процесам та, за необхідності, реорганізації таких структур даних є складною задачею. Складність вирішення цієї задачі полягає, зокрема, у застосуванні в автоматизованих системах відмінних технологій збереження, доступу та обробки даних з намаганням отримати найкращі показники ефективності для кожного з процесів, при цьому загальна ефективність системи, що базується на таких технологіях, часто виявляється не найкращою. Так, для збереження даних найбільше поширення отримали системи з реляційними базами даних, для обробки даних — програмне забезпечення, що базується на технологіях об'єктно-орієнтованого програмування. Як наслідок, в системах, що базуються на таких технологіях виникає невідповідність організації даних у сховищі (кортежі, пов'язані асоціативними зв'язками), та у додатках (об'єкти, пов'язані відношеннями агрегації, асоціації та залежності) з часом зростає складність структур даних, що, в цілому, призводить до необхідності побудови і виконання над базами даних вкрай складних запитів та ускладнює реорганізацію цих структур для збереження прийнятної ефективності програмного забезпечення та сховищ даних.

Складність та різноплановість процесів створення баз даних та їх істотна залежність від бізнес логіки компанії зайвий раз підтверджується значною кількістю методологій проектування бізнес-процесів та організації баз даних. Як наслідок, підхід до проектування програмного забезпечення АІС та баз даних, що базується на намаганні отримати найбільш ефективні рішення

розробки програмного забезпечення та організації бази даних системи призводять до зниження ефективності системи в цілому. Так, невиправдане використання найбільш поширених реляційних баз даних призводить до невідповідності функціоналу, що забезпечується СКБД, вимогам до механізмів доступу, обробки і аналізу даних, які визначаються програмним забезпеченням на базі технологій об'єктно-орієнтованого програмування зокрема та бізнес-логікою в цілому. При цьому розробка запитів для підготовки даних для бізнес-аналізу на складних структурах даних є невиправдано трудомістким завданням і, в окремих випадках, вимагає створення допоміжних таблиць даних (Temporary tables) або видів (Views), що, у свою чергу, додатково ускладнює роботу з базою даних через зростання вимог до обсягів оперативної пам'яті. Крім того, будь-які зміни структури бази даних внаслідок змін у бізнес логіці або розвитку систем змушує розробників здійснювати переробку раніше створених запитів і розробленого програмного забезпечення у цілому.

Вважається, що серед різних компонентів апаратури сервера СКБД конфігурація пам'яті системи зазвичай має найбільше вплив на її продуктивності — більша частина основної пам'яті в системах зберігання і обробки даних використовується як буфер (кеш) для тимчасового збереження даних, що в багатьох випадках дозволяє усунути необхідність виконання операцій фізичного введення-виведення з диска. Оскільки звернення до основної пам'яті виконується приблизно в 10000-100000 разів швидше, ніж звернення до швидкого диску, мінімізація дискового вводу-виводу є першочергово важливим завданням при вирішенні задач підвищення ефективності сховища даних.

Існуюче в даний час співвідношення між цінами підсистеми пам'яті і дискової підсистеми показує, що є економічно доцільним кешувати дані, до яких здійснюються звернення більше одного разу на кожні декілька хвилин роботи системи. Це означає, що дані, до яких звернення відбуваються більш часто, ніж один раз на декілька хвилин, повинні кешуватися в оперативній

пам'яті. Відповідно, щоб оцінити розмір кеша даних необхідно підсумувати об'єми всіх даних, які мають використовуватись більш часто, ніж один раз на декілька хвилин на рівні всієї системи. Тому при визначенні необхідного обсягу основної пам'яті розраховують, принаймні, на такий розмір кешу даних. Додатково зазвичай резервують ще 5-10% для розміщення лужбової інформації про структуру і організацію даних (верхніх рівнів індексів В-дерев, типів даних), збережених процедур та іншої керуючої та допоміжної інформації СКБД.

Механізми аналізу ефективності більшості сучасних комерційних СКБД забезпечують можливості визначення вартості чи переваг зміни розміру різних кешів СКБД. Ці механізми дозволяють визначити ефект від зміни розмірів кешу вводу-виводу СКБД. На жаль, ці механізми не дозволяють визначити вплив структури даних та організації даних у базі даних на ефективність системи.

Природно, конфігурація системи повинна забезпечувати також простір для традиційного використання пам'яті. В СКБД на базі Unix і MS Windows завжди необхідно забезпечити принаймні 100-200 Мбайт для ядра базової операційної системи та системних служб. Далі, необхідно передбачити окремий простір для ряду програм бізнес-аналітики на рівні СКБД а також програм обслуговування сховища даних: ведення журналу, перевірки узгодженого стану, архіваторів і т. п.

Оброблювана кожним процесором додаткова інформація вимагає виділення деякого простору для того, щоб обійти інтенсивну фрагментацію пам'яті на зовнішніх носіях. Ці задачі виявляються тим більше складними, що помилки вибору структури бази даних важко виправляти через істотні розбіжності в механізмах організації даних в базах даних.

Наслідком вище зазначеного є обмеження на обробку даних в системах збереження даних, коли обробка базується на всіх або значній кількості даних — така обробка потребуватиме обсягів оперативної пам'яті рівних або навіть більше, ніж обсяг даних у БД, який в бізнес секторі може становити одиниці або

навіть сотні терабайт. Саме тому така обробка даних потребує обсягів обчислювальних ресурсів, які є недосяжними в класичних сховищах даних і потребує спеціальних механізмів. Модель доступу до класичного сховища даних корпоративної інформаційної системи (пасивному, без можливостей використання результатів аналізу структури та аналізу бізнес-процесів) представлена на рисунку 1.1.

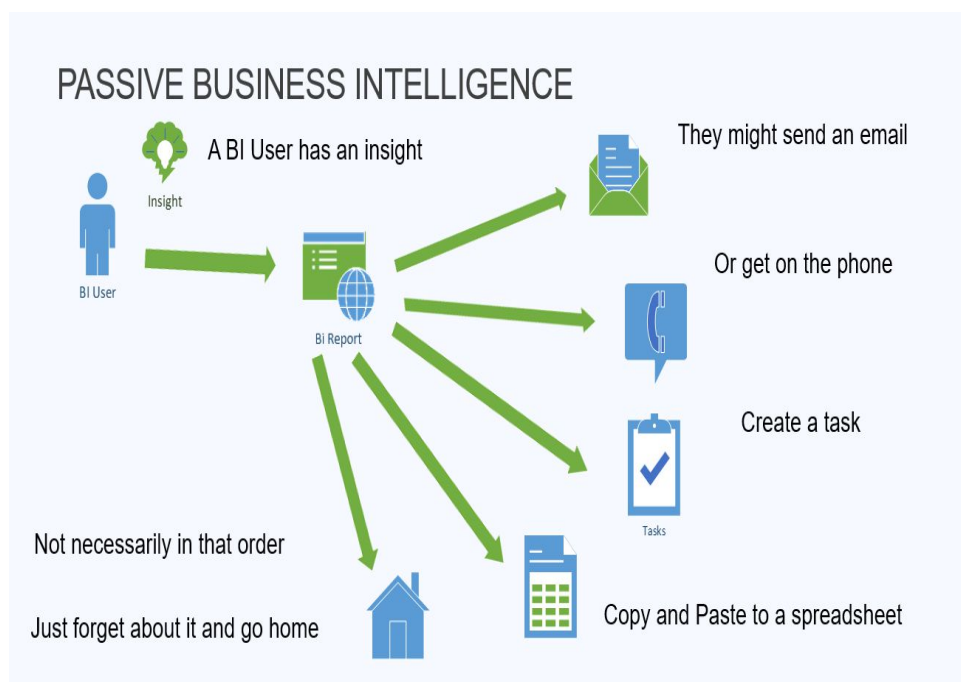


Рис. 1.1. Модель доступу до даних у пасивному сховищі даних корпоративної БД

Для вирішення задач бізнес-аналізу даних традиційно використовують системи онлайнової аналітичної обробки (OLAP) – це технологія, яка використовується для організації великих бізнес-баз даних і підтримки бізнес-аналітики. Бази даних OLAP розділено на один або кілька кубів, кожен із яких упорядковано та розроблено адміністратором куба відповідно до способу отримання і аналізу даних, що спрощує створення й використання звітів зведених таблиць і зведена діаграма потрібних звітів.

З метою визначення особливостей реалізації бізнес логіки в корпоративних

інформаційних системах та її відображення в моделі та механізми збереження та доступу до даних для СКБД розглянемо особливості основних типів сучасних баз даних та визначимо їх переваги та недоліки стосовно розробки навантажених систем збереження даних та їх бізнес-аналізу.

1.1.2. Реляційні СКБД

Корпоративні рішення на базі реляційних СКБД є традиційними для багатьох проєктів через поширення цих СКБД та високий рівень формалізації реляційної алгебри. Вони також дозволяють забезпечити відображення даних предметної області в структури даних сховища у найбільш простий спосіб (Рис. 1.2).

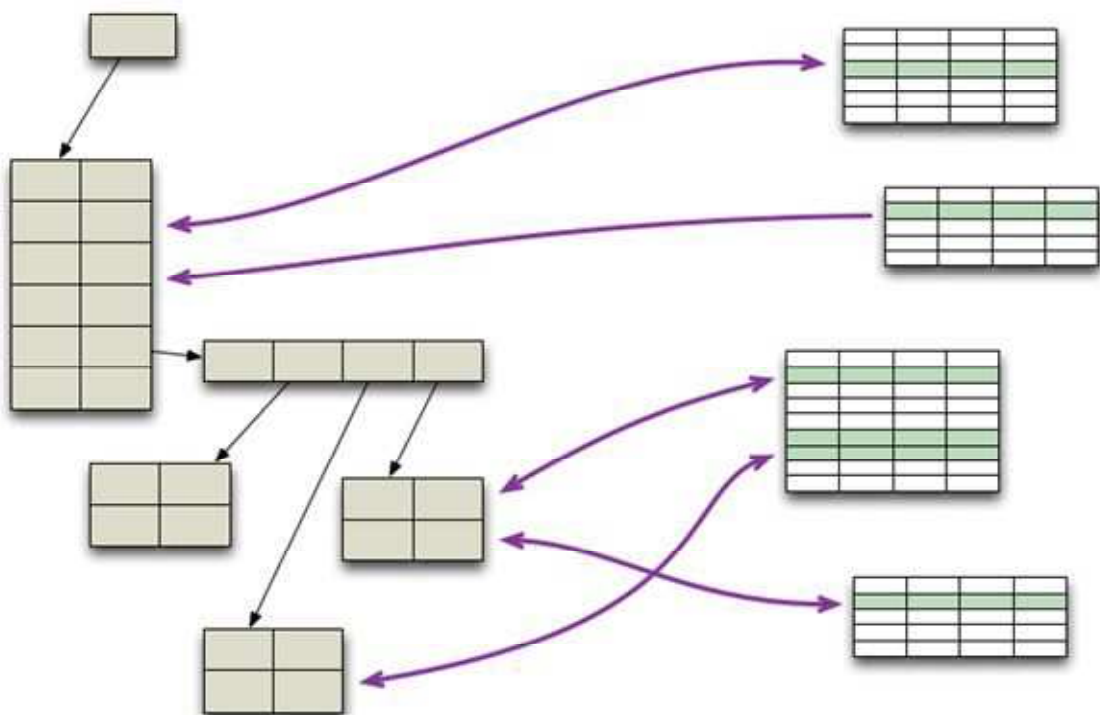


Рис. 1.2. Відображення бази даних (праворуч) на предметну область (ліворуч) в реалізації інформаційної системи для РСКБД

Найбільш поширеними базами даних є наступні реляційні – MySQL, PostgreSQL, MSSQL і Oracle. Перші дві безкоштовні і найчастіше використовуються для корпоративних систем з інтернет доступом. Другі дві – платні і рідше використовуються в корпоративних системах і, зокрема, проєкт сховища даних корпоративної інформаційної системиах, а частіше в корпоративних інформаційних системах з бізнес-аналітикою. По суті, для звичайних проєктів в технічному плані немає істотної різниці, яку СКБД використовувати, але в економічному плані вигідніше використовувати найпоширенішу MySQL (та її відгалудження MariaDB) або трохи менш поширену PostgreSQL, для яких більше розробників, нижче вартість підтримки і розробки.

MariaDB – відгалудження реляційної СКБД MySQL, розроблювальне сообществом під ліцензією GPL. MariaDB повністю сумісна з додатками, що використовують MySQL, а перехід на цю СКБД виправданий тим, що MySQL вже не так активно розвивається.

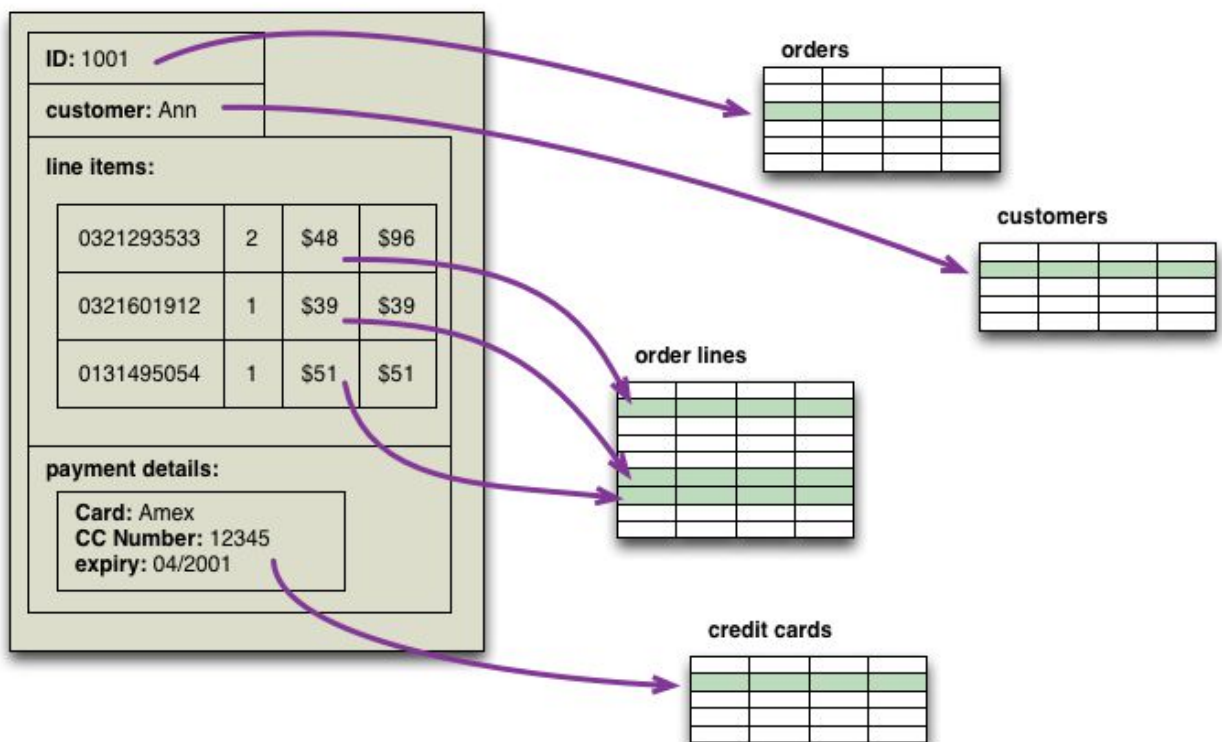


Рис. 1.3. Відображення предметної області в реалізації інформаційної

системи для РСКБД

Крім зрозумілої політики поширення продукту, дана СКБД відрізняється від MySQL більш високою продуктивністю, новими можливостями і меншою кількістю помилок в коді (зокрема, за рахунок кращого покриття тестами і більш активного розвитку).

У MariaDB вбудовані покращений оптимізатор запитів, безпечна і швидка реплікація, більш швидкі індекси для механізму зберігання даних MEMORY, більш висока продуктивність перекодування символів, використання пулу потоків, а також багато інших покращання, позитивно впливають на продуктивність.

У MariaDB відбулася відмова від підсистеми зберігання даних InnoDB і його заміна на XtraDB. Також на додаток до MyISAM, Blackhole, CSV, Memory і Archive включені підсистеми зберігання Aria, PBXT (до версії 5.5), FederatedX, OQGRAPH, SphinxSE, IBMDB2I (до версії 5.5), TokuDB, а також з версії 10.0 – Cassandra, CONNECT, SEQUENCE, Spider.

1.1.3. NoSQL СКБД

Останнім часом для вирішення завдань, пов'язаних зі складними методами обробки даних і складною організацією даних (що характерно при автоматизації складних бізнес процесів) набули поширення так звані NoSQL бази даних та сховища (MongoDB, CouchDB, Redis, Memcached). Вони значно молодше реляційних баз даних, істотно відрізняються від них і застосовуються частіше не для зберігання всіх даних системи, а лише для вирішення специфічних завдань (журналювання, кешування, черги завдань), і тому менш поширені в простих проектах.

MongoDB – це NoSQL сховище даних, вкрай зручне для зберігання інформації, яка не може бути у простий спосіб нормально структурована в рамках реляційних баз даних.

MongoDB – СКБД з відкритим вихідним кодом, що не вимагає опису схеми таблиць і написана на мові C++. Завдяки спеціалізації бази даних вдалося відійти від принципу «один розмір під все», а за рахунок мінімізації методів роботи з транзакціями з'явилася можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, що також сприяло поліпшенню горизонтального масштабування.

Документи (дані) в MongoDB зберігаються в JSON або BSON форматах (рис 1.5), робота з такою моделлю простіше кодується і простіше управляється, а внутрішня угруповання релевантних даних забезпечує додатковий виграш у швидкодії. MongoDB, за даними розробників, повинна заповнити розрив між найпростішими NoSQL-СКБД, що зберігають дані у вигляді «ключ – значення» (простими і легко масштабованими, але володіють мінімальними функціональними можливостями, такими як Memcached, наприклад) і великими реляційними СКБД (зі складною організацій даних (структурними схемами даних) і системами генерації складних запитам з вбудованими оптимізаторами).

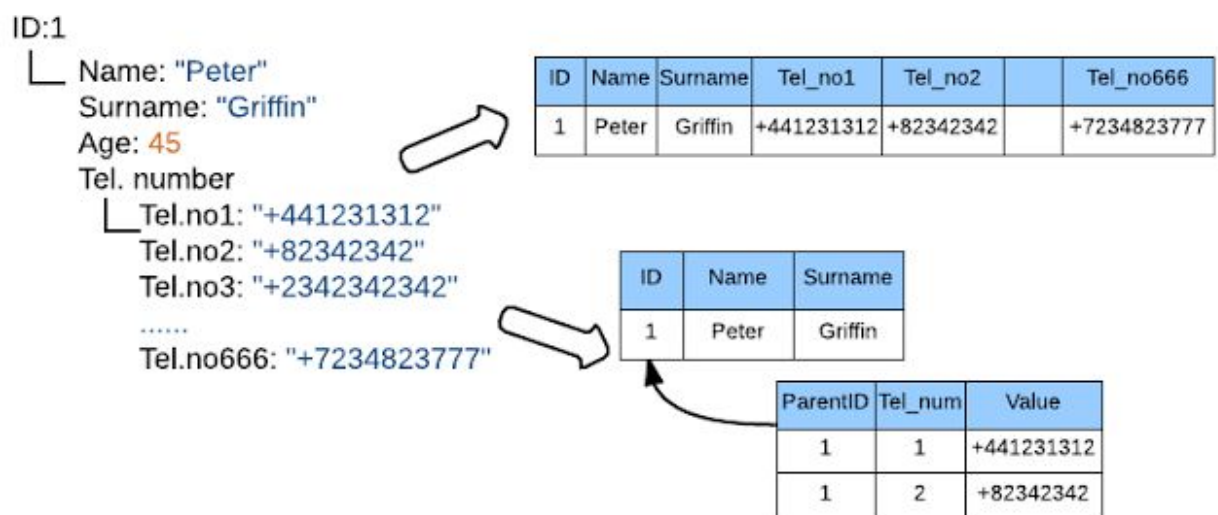


Рис. 1.4. Відображення предметної області в реалізації інформаційної системи для NoSQL СКБД

Переваги NoSQL:

- гнучка мова для формування запитів;
- динамічні запити;
- індексація колекцій;
- профілювання запитів;
- журналювання операцій запису;
- відмовостійкість і масштабованість;
- асинхронна реплікація і шардінг;
- повнотекстовий пошук з підтримкою морфології.

Робота з MongoDB реалізована в мовах програмування Java, C++, C#, PHP, Python, Perl, Ruby та інших, а відповідні компоненти є в багатьох фреймворках (наприклад, в Ruby on Rails або в Yii).

1.1.4. Об'єктно-орієнтовані СКБД

Зазначена проблема могла б бути подолана за рахунок використання об'єктно-орієнтованих систем керування базами даних, що дозволяють повною мірою задовольнити потреби в організації даних з боку об'єктів програмного забезпечення, але, на жаль, об'єктно-орієнтовані СКБД на сьогодні не набули широкого поширення, насамперед – через свою складності і надмірної вартості. Перші публікації були викликані проблема РСКБД при високій складності відношень (рис. 1.1).

Object 1: Maintenance Report**Object 1 Instance**

Date	
Activity Code	
Route No.	
Daily Production	
Equipment Hours	
Labor Hours	

01-12-01
24
I-95
2.5
6.0
6.0

Object 2: Maintenance Activity

Activity Code	
Activity Name	
Production Unit	
Average Daily Production Rate	

ис.
1.5.
Від
обр
аже
ння
дан
их
в
ОО
СК
БД

Об'єктно-орієнтована база даних (ООБД) зазвичай рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, що мають складну структуру.

Обов'язкові характеристики, яким має відповідати будь-яка ООБД наступні:

- Підтримка складних об'єктів. В системі має бути передбачена можливість створення складових об'єктів за рахунок застосування конструкторів. Необхідно, щоб будь-який з конструкторів можна було використовувати з будь-яким об'єктом.
- Підтримка індивідуальності об'єктів. Всі об'єкти мають мати унікальний ідентифікатор, котрий не залежить від значень їх атрибутів.
- Підтримка інкапсуляції. Інкапсуляція досягається за рахунок того, що програмісти мають право доступу тільки до специфікації інтерфейсу методів, а дані і реалізація методів приховані всередині об'єктів.
- Підтримка типів і класів. Вимагається, щоб ООБД підтримувалась хоча б

одна концепція відмінності між типами і класами.

- Підтримка наслідування типів і класів від їх предків. Підтип або підклас має наслідувати атрибути і методи від супертипу або суперкласу.
- Перезавантаження у поєднанні з повним пов'язуванням. Реалізація методу має залежати від типу об'єктів, для якого даний метод викликається.
- Обчислювальна повнота. Мова операцій над даними має бути мовою програмування загального призначення.
- Набір типів даних має бути розширюваним. Користувач повинен мати засоби створення нових типів даних на основі набору зумовлених системних типів. Більш того, між способами використання системних і користувацьких типів даних не повинно бути жодних відмінностей.

З метою втілення зазначених концепцій та подолання наявних в РСКБД проблем останнім часом широко використовують об'єктно-орієнтовані надбудови над реляційними базами даних (ORM, Object Relational Mapping), які забезпечують підтримку відповідності організації даних на рівні додатку і на рівні структур даних бази даних. Таким чином, використання ORM надає розробнику можливість відносно вільно маніпулювати даними на рівні об'єктів і абстрагуватися від особливостей реалізації структур даних на конкретному сервері баз даних. Це також дозволяє відносно простими засобами отримувати зрозумілий і якісний код, спрощує його тестування, подальшу модифікацію і супровід.

Деякі програмні пакети вирішують цю проблему, надаючи бібліотеки класів, здатних виконувати перетворення даних автоматично. Маючи список таблиць в базі даних і об'єктів в програмі, вони автоматично перетворюють об'єктні запити в реляційні та забезпечують конвертацію даних з внутрішнього подання бази даних в об'єктне уявлення, придатний для використання в об'єктно-орієнтованій програмі. При цьому в результаті виконання ORM-запиту генеруються і виконуються відповідні запити до таблиць сервера баз даних, а результати перетворюються в об'єкти відповідного типу.

ORM також відмінно підходить для реалізації і втілення такої сучасної парадигми ОО-програмування, як MVC (Model-View-Controller, рис 1.6). Ця парадигма дозволяє відокремити інтерфейси програми від даних і бізнес логіки, а також загальних операцій над даними (рис. 1.7).

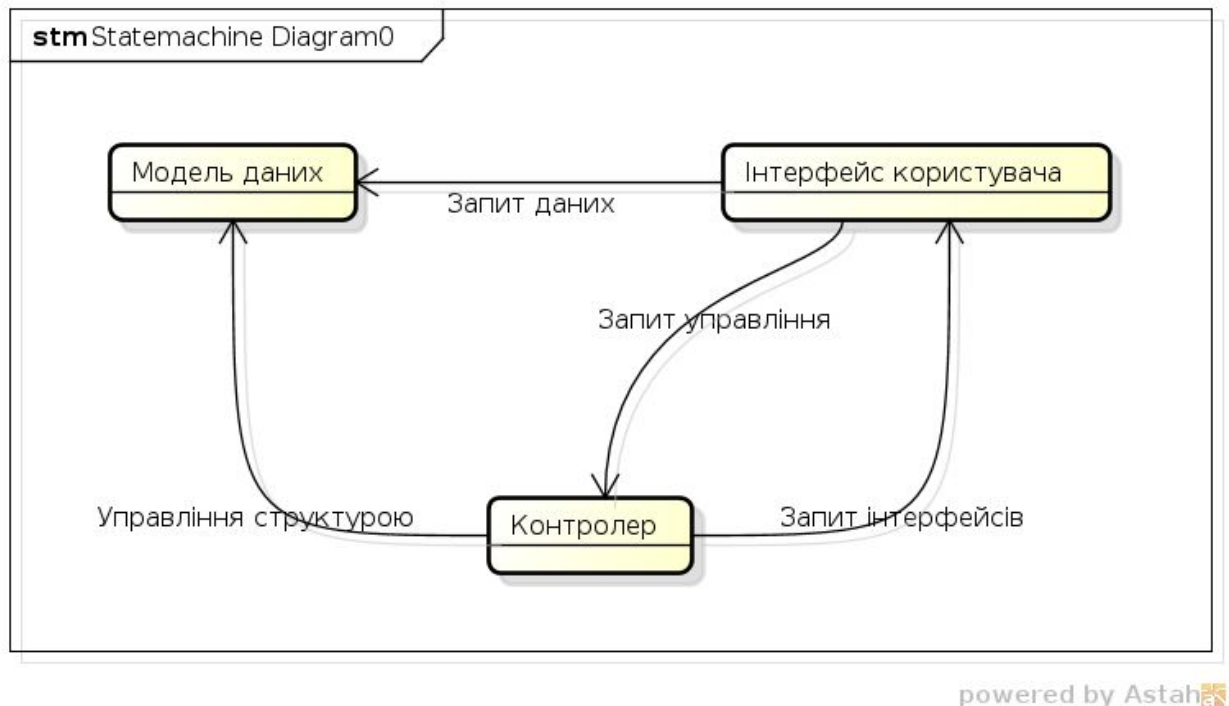


Рис. 1.6. Структура додатку для моделі Model-View-Contriller

Застосування ORM-технологій має і свої недоліки, насамперед, через деяке зменшення пропускної здатності сервера баз даних. Більше того, шар транзакцій при цьому виявляється повільнішим, а часто ще й неефективним (особливо в термінах згенерованого запиту, записаного мовою запитів – SQL). Таким чином, використання ORM може призвести до того, що програми, які його використовують, будуть працювати повільніше і використовувати більше оперативної пам'яті, ніж програми, що містять «вручну» написані запити. Тим не менше, ORM позбавляє розробника від необхідності написання великої кількості коду, часто одноманітного і схильного до помилок, і, головне, який тісно пов'язаний зі структурами даних, що унеможлиблює оперативну модифікацію цих структур. Тим самим ORM значно підвищуючи швидкість

розробки. На додаток до можливостей генерації запитів на основі об'єктних моделей більшість сучасних реалізацій ORM дозволяють також жорстко визначати код SQL запитів, який буде використовуватися при тих чи інших непередбачених розробниками ORM діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

PureMVC – це фреймворк, що базується на шаблонах проектування. Він виник з нагальної необхідності проектування високопродуктивних RIA-клієнтів. Зараз він вже портований іншими мовами та платформами, включаючи серверні середовища[хоча інтерпретація та реалізація мають свої особливості для кожної підтримуваної PureMVC мовних платформ.

Взагалі, для проектів, які характеризуються високим навантаженням на сховища даних, питання отримання необхідної продуктивності гостро стоїть і без застосування ORM. Але для таких систем також мають велике значення та можливості отримання відносно простих інструментів розширення системи, а саме такі можливості і забезпечуються використанням ORM.

Впровадження ORM зазвичай призводить до деградації продуктивності системи на близько 30%. Маючи на увазі швидкий прогрес у підвищенні продуктивності обчислювальних систем (близько 30% – 50% на рік) і тенденції до зменшення вартості обладнання, можна розраховувати на те, що протягом року недоліки системи, пов'язані з деградацією обчислень внаслідок використання ORM будуть повністю нівельовані за рахунок підвищення продуктивності апаратного забезпечення. При цьому переваги від впровадження ORM, пов'язані з можливостями більш простого розширення структур даних, повною мірою будуть збережені і надалі.

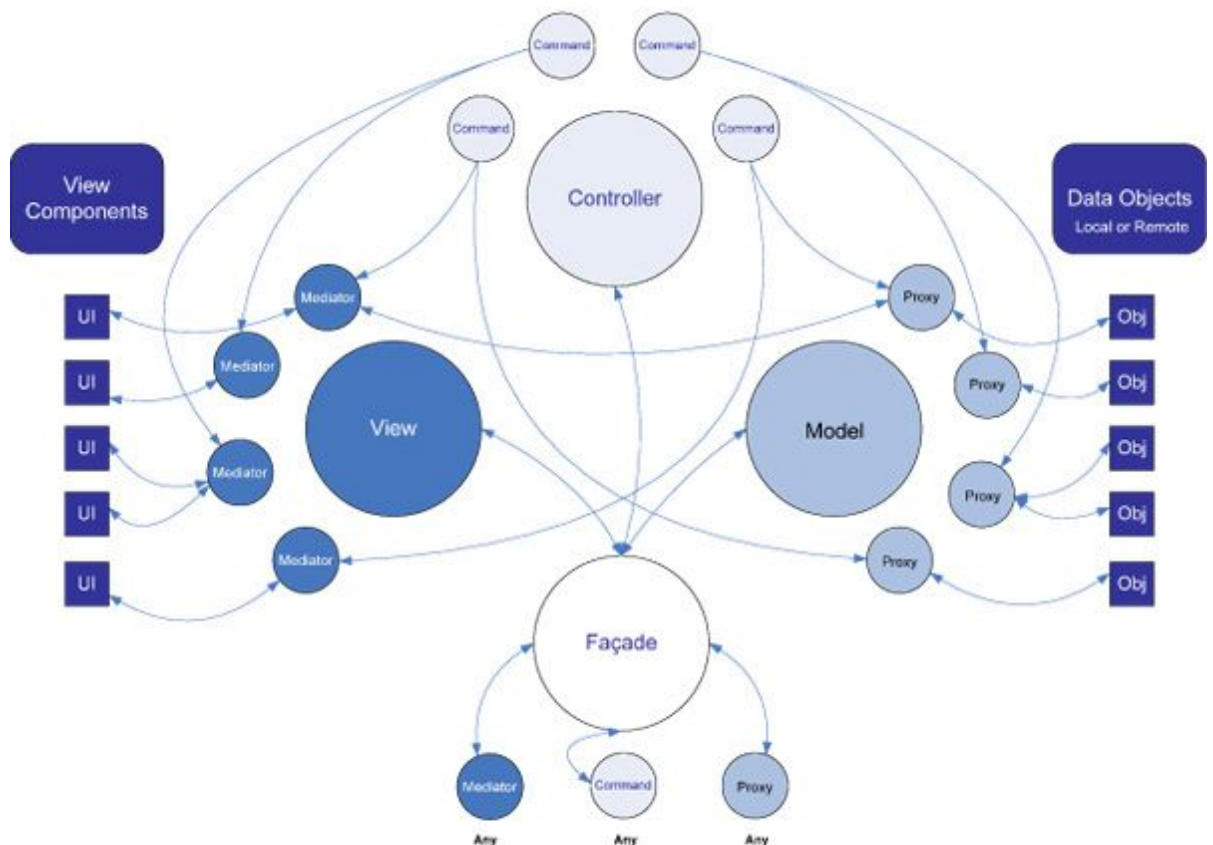


Рис. 1.7. Деталізована структура додатку в моделі Model-View-Contriller для платформ PureMVC

1.1.5. Документо-орієнтовані бази даних

Документо-орієнтовані бази даних — спеціально призначені для зберігання ієрархічних структур даних (документів) і зазвичай реалізовані за допомогою підходу NoSQL. В основі документо-орієнтованих СКБД лежать документні сховища, які мають структуру дерева або лісу. Структура дерева починається з кореневого вузла і може містити кілька внутрішніх і листових вузлів. Листові вузли містять дані, які при додаванні документа заносяться в індекси, що дозволяє навіть при досить складній структурі знаходити місце (шлях) шуканих даних.

API для пошуку дозволяє знаходити в документо-орієнтованих базах даних за запитом документи або окремі частини документів. На відміну від сховищ типу ключ-значення, вибірка за запитом до документного сховища може містити

частини великої кількості документів без повного завантаження цих документів на клієнтські машини. Такий підхід виявляється цінним для систем, в яких обсяг трафіку між клієнтами та сервером є критичним.

1.1.6. Розподілені бази даних

Розподілені БД складають ще один напрям в просторі досліджень і розробок систем керування базами даних. У цих системах доводиться вирішувати всі завдання, властиві централізованим СКБД, але, як правило, в більш складних постановках. У розподілених системах виникають і специфічні проблеми, від вирішення яких багато в чому залежить ефективність, надійність і доступність систем БД. В даний час більшість розподілених СКБД базується на реляційній моделі даних і розрахована на використання в локальних обчислювальних мережах. Багато проблем поширюються і на розподілені СКБД в територіально рознесених мережах, і майже всі проблеми зберігаються для розподілених СКБД, заснованих на інших моделях даних.

Розподілена база даних (РБД) — це множина логічно взаємозалежних баз даних, розподілених у комп'ютерній мережі.

Серед особливостей використання таких баз даних можна відзначити:

- великий обсяг обмінів даними (напружений мережевий трафік);
- низьку надійність системи через відмову каналів зв'язку та центральної СКБД;
- низьку загальну продуктивність (вузькі місця — мережа і центральний сервер);
- великі витрати на розробку і експлуатацію.

1.1.7. Висновки щодо методів та засобів збереження даних

Для звичайних проектів простіше використовувати MySQL або PostgreSQL, оскільки на простих операціях не надто помітна різниця між різними базами даних. Однак, якщо проект передбачає складну логіку обробки даних, то вибір

бази варто проводити виходячи з технічних характеристик та області використання кожної окремої системи та її сховища даних (Таблиця 1.1, рис 1.8).

1.2. Огляд інтернет-сервісів та протоколів доступу до даних корпоративних інформаційних систем

1.2.1. Механізми доступу та сервіси

Окрім структури організації даних бази даних також істотно відрізняються механізмами доступу, які можуть бути реалізовані на різних рівнях реалізації інформаційної системи: на рівні файлової системи ОС, СКБД або навіть на рівні додатку.

Залежно від рівня реалізації механізмів доступу також істотно відрізняються можливості щодо побудови складних запитів (чим нижче рівень – тим складніше) та швидкості їх виконання (чим вище рівень – тим повільніше).

Таблиця 1.1. Типи сховищ баз даних за областями призначення

<i>№</i>	<i>Тип сховища</i>	<i>Область використання сховища</i>
	Реляційні СКБД	Класичні проекти з відносно простою бізнес логікою
	NoSQL СКБД	Проекти зі складною бізнес логікою та відносно невеликим обсягом даних
	ОО СКБД	Проекти зі складною бізнес логікою та відносно великим обсягом даних
	Документо-орієнтовані СКБД	Проекти з ієрархічно-впорядкованими даними
	Розподілені СКБД	Проекти систем обробки даних з низькими вимогами до надійності

	Файлові сховища	Проекти з масовим доступом до окремих файлів
--	-----------------	--

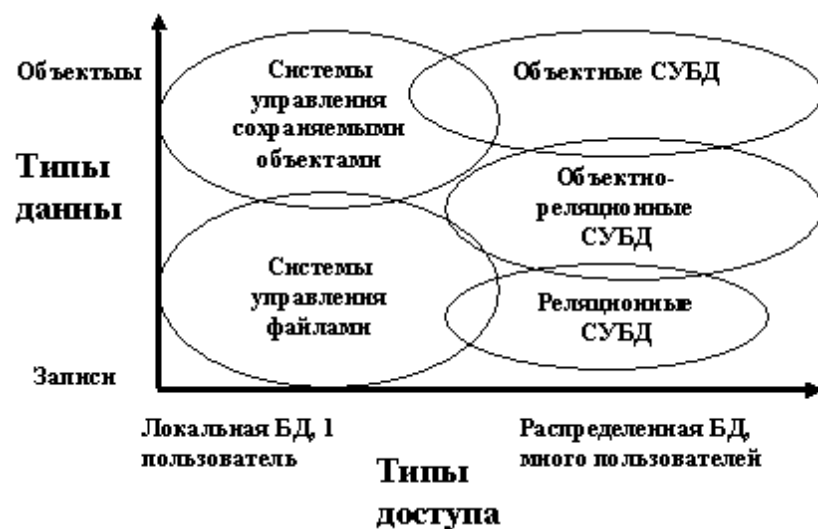


Рис. 1.8. Области використання СКБД з різною організацією

1.2.2. Інтернет-сервери HTTP та HTTPS

Найбільш популярний серед інтернет-розробників вважається комплект LAMP (Linux, Apache, MySQL, PHP). Цей інтернет-сервер дозволяє підключати зовнішні модулі для роботи з інтерпретуються мовами програмування (наприклад, з PHP, Ruby, Python) і для надання даних, може використовувати СКБД для аутентифікації користувачів, можлива модифікація повідомлень про помилки і багато іншого. Apache є кросплатформним ПЗ, підтримує операційні системи Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS, хоча найбільш часто використовується тільки на серверних дистрибутивах Linux і BSD.

Сам по собі інтернет-сервер вельми потужний і надійний, але споживання ресурсів при його роботі зазвичай досить високо. Часто Apache використовують як бекенд-сервера, встановлюючи позаду інтернет-сервера nginx.

HTTPS – захищений протокол передачі гіпертексту. HTTPS – розширення

протоколу HTTP, що підтримує шифрування. Передані по протоколу HTTPS дані зашифровані, що забезпечує захист від прослуховування. HTTPS широко використовується в світі і підтримується всіма популярними браузерами.

HTTPS (HyperText Transfer Protocol Secure) – це не окремий протокол, а розширення звичайного HTTP, що працює через шифровані транспортні механізми SSL або TLS. HTTPS забезпечує захист від атак, заснованих на прослуховуванні мережевого з'єднання, тобто від сніфферських атак і атак типу man-in-the-middle.

На відміну від HTTP, який використовує TCP-порт 80, для HTTPS за замовчуванням використовується TCP-порт 443. Щоб підготувати інтернет-сервер для обробки https-з'єднань, адміністратор повинен отримати і встановити в систему SSL-сертифікат для цього інтернет-сервера.

SSL-сертифікат – це свого роду «електронний паспорт» сайту, в якому він підтверджує, що він дійсно той сайт, за який видає себе. Сертифікат сайту покликаний додатково підтверджувати, що цей сайт – дійсно сайт банку, платіжної системи і т.д. SSL-сертифікат, як і всякий паспорт, зберігає різні дані: найменування організації та підрозділи, країну, регіон, місто і ім'я сервера, для якого даний сертифікат був створений.

Також, для нормальної роботи HTTPS, SSL-сертифікат повинен бути підписаний організацією, якій довіряють все. У разі SSL-сертифікатів такими організаціями є центри сертифікації. І якщо сертифікат підписаний надійним центром сертифікації, то вважається, що він містить правильну і відповідну дійсності інформацію.

1.2.3. Особливості використання файлових сховищ в задачах бізнес-аналітики

Файлові сховища або файловий хостинг – сервіс, що надає користувачу місце під його файли і цілодобовий доступ до них через інтернет-інтерфейс, як правило за протоколом HTTP. Такий сервіс дозволяє зручно «обмінюватися»

файлами. На спеціальній сторінці файлового сховища (найчастіше на головній) користувач завантажує файл на сервер сховища і отримує постійне посилання, яке він може розсилати по e-mail, публікувати в блогах, на форумах тощо. Перейшовши за таким посиланням інший користувач може викачати розміщений файл.

Інформація в файлових сховищах поділена, зазвичай, на три базові категорії:

- **Захищена інформація.** Режим доступу визначається власниками і надається за спеціальною угодою. До цього виду ресурсів відносяться комерційні архіви (наприклад, комерційні версії програм в архівах <ftp.microsoft.com>), закриті національні і міжнародні некомерційні ресурси (наприклад, роботи за міжнародними проектами CES або IAEA), приватна некомерційна інформація із спеціальними режимами доступу (наприклад, приватні добродійні фонди).
- **Інформаційні ресурси обмеженого використання.** До них відносяться ресурси обмеженого часу використання або обмеженого часу дії, наприклад, програми класу shareware.
- **Відкриті інформаційні ресурси.** До цих ресурсів відноситься все, що можна вільно отримати по мережі без спеціальної реєстрації (freeware). Це може бути документація, безкоштовне програмне забезпечення, фотографії, звукові файли тощо. Зазвичай, вільно поширюване програмне забезпечення не має сертифікату якості, але його розробники є відкритими для обміну досвідом.

Найбільшої популярності файлові хостинги набули, коли зростання користувачів Інтернету, можливості розміщення чи завантаження файлів великого розміру почали спричиняти величезного навантаження на багато ресурсів.

Завдяки файловим хостингам вдалося вирішити проблему перевантаження для ряду ресурсів, а також надати для звичайного користувача

можливість передавання файлу, наприклад, розміром 100 мегабайт абсолютно безкоштовно. Природно, що за безкоштовність доводиться платити (банери на сторінці завантаження, відтерміноване викачування користувачем, примусовий перегляд інтернет-сторінок), бо витрати власників файлових хостингів необхідно якось окупати.

Перш ніж завантажити на певний сервіс файли корпоративної системи, слід ознайомитися з правилами роботи служби та файлового сховища. Також необхідно зважати, що одна з головних вимог публічних сховищ – обмеження щодо нелегального контенту (зокрема, неліцензійних програм, піратських копій медійного забезпечення тощо), використання якого порушує авторські права чи може вплинути на працездатність сервера. Якщо подібний матеріал і з'являється, він оперативно видаляється адміністрацією, а осіб, що його завантажили можуть притягнути до юридичної відповідальності. Ці обмеження діють навіть у випадках, якщо збереження таких копій (наприклад – комп'ютерних вірусів) є аспектом легальної діяльності підприємства, яке завантажує файли на сервер.

Більшість файлових сховищ дозволяють викачувати файли лише одним потоком, по одному за раз, і часто з паузою в годину/дві перед завантаженням наступного файлу. Пам'ятати, що вже викачано, а що ні, скільки часу лишилося до можливості нового викачування, отримати пряме посилання на файли, щоб поставити їх в чергу менеджера закачування – досить складно та непродуктивно. Кращим вирішенням цього обмеження є застосування спеціальних програм для викачування з файлових сховищ, наприклад, розробка Universal Share Downloader.

Для доступу до файлових сховищ здебільшого використовують спеціалізовані протоколи передачі даних. SFTP – протокол передачі файлів прикладного рівня, призначений для копіювання та виконання інших операцій з файлами поверх надійного і безпечного з'єднання. SSH використовується для безпечної передачі, що приховує (шифрувальної) логін і пароль, а також

шифрувальної вміст.

SFTP не пов'язаний з FTP, за винятком того, що він теж передає файли і має аналогічний набір команд для користувачів. SFTP, або безпечний FTP, – це програма, що використовує SSH (Secure Shell) для передачі файлів. На відміну від стандартного FTP він шифрує і команди, і дані, оберігаючи паролі та конфіденційну інформацію від відкритої передачі через мережу. За функціональністю SFTP схожий на FTP, але так як він використовує інший протокол, клієнти стандартного FTP не можуть зв'язатися з SFTP-сервером і навпаки.

Використання таких протоколів, як і файлових сховищ доцільне лише у випадку розподілених сховищ даних, коли дані зберігаються на одних вузлах розподіленої системи, а бізнес-аналітика виконується на інших, як правило спеціалізованих, вузлах системи.

1.2.4. Задачі бізнес-аналітики і технологія OLAP

OLAP – зручний інструмент аналізу, перегляду і візуалізації інформації. Традиційні звіти, навіть побудовані на основі єдиного сховища, позбавлені одного – гнучкості. Їх не можна "прокрутити", "розгорнути" або "згорнути", щоб отримати бажане представлення даних, тому отримання даних, їх представлення є доволі тривалою задачею. Як наслідок, бізнес-аналітик звичайно може перевірити за день декілька ідей, що не відповідає вимогам вирішення задач бізнес-аналітики, які мають вирішуватись онлайн поротягом одиниць секунд.

Оперативні для OLAP дані збираються з різних джерел, очищаються, інтегруються та складаються у сховище (звичайно — реляційне). При цьому вони вже доступні для аналізу за допомогою різноманітних засобів побудови звітів. Далі дані (всі повністю або частково) готуються для OLAP-аналізу. Вони можуть бути завантажені у спеціальну БД OLAP або залишені у основному сховищі. Найважливішим його елементом є метадані, тобто інформація про

структуру, розміщення та трансформацію даних. Завдяки їм забезпечується ефективна взаємодія різних компонентів сховища. Підсумовуючи, можна визначити OLAP як сукупність засобів багатовимірного аналізу даних, накопичених у сховищі. Теоретично засоби OLAP можна застосовувати безпосередньо до оперативних даних або їх точним копіям (щоб не заважати оперативним користувачам). Але ми цим ризикуємо наступити вже описані вище граблі, т. е. почати аналізувати оперативні дані, які безпосередньо для аналізу непридатні.

Користувач OLAP отримує природну, інтуїтивно зрозумілу модель даних, організуючи їх як багатомірні куби (Cubes). Куб OLAP надає зручні швидкі засоби доступу, перегляду та аналізу ділової інформації. Осями багатовимірної системи координат є основні атрибути аналізованого бізнес-процесу. Наприклад, для продажу це може бути товар, регіон, тип покупця. Як один з вимірів використовується час. На перетинах осей – вимірювань (Dimensions) – знаходяться дані, що кількісно характеризують процес – заходи (Measures). Це можуть бути обсяги продажу в штуках або в грошовому вираженні, залишки на складі, витрати і т.п. по тижнях) відомості та здійснювати інші маніпуляції, які йому спадають на думку в процесі аналізу. Як заходи у тривимірному кубі, використані суми продажів, а як вимірювання — час, товар і магазин. Вимірювання представлені певних рівнях угруповання: товари групуються за категоріями, магазини — країнами, а дані час проведення операцій — за місяцями. Трохи згодом ми розглянемо рівні угруповання (ієрархії) докладніше.

1.2.5. Висновки щодо методів та засобів передачі даних

Широкий спектр технологій доступу до даних породжує проблеми вибору цих технологій в окремих проектах (Таблиця 1.2).

Вибір конкретних технологій здійснюється, як правило, за результатами вибору бази даних для проекту.

Таблиця 1.2. Типи технологій доступу залежно від баз даних

№	Тип доступу	Область призначення сховища
1	HTTP-доступ	Файлові сховища, реляційні СКБД
2	FTP-доступ	Файлові сховища з розподіленим зберіганням
3	P2P-доступ	Файлові сховища з розподіленим зберіганням

1.3. Проблеми доступу до даних і аналізу даних в проектах корпоративних інформаційних систем

1.3.1. Проблеми захисту даних та каналів передачі даних

Очевидно, що засоби захисту даних від несанкціонованого доступу дуже важливі і будуть продовжувати вдосконалюватися. Будуть ускладнюватися алгоритми кодування даних на всіх рівнях (в БД, архівах, при передачі по мережі), способи авторизації і аутентифікації і т.д. Останнім часом з'явилася тенденція виносу механізму управління користувачами (обліковими записами) з окремих СКБД і додатків в єдину централізовану систему організації (Identity & Access management). Що спрощує управління користувачами в масштабах організації та дозволяє керувати ними з урахуванням вимог бізнесу, в автоматичному режимі. Удосконалюються методи аудиту, з'являтиметься інтегрована середовище збору та аналізу аудит-інформації всієї організації (зі безлічі додатків і БД).

Крім того, інтенсивно вдосконалюються засоби захисту даних усередині БД. Наприклад, швидко розвивається механізм завдання політик перевизначення запитів на льоту, тобто запит автоматично модифікується залежно від самих різних параметрів (назва програми, час запуску на виконання, ім'я користувача, місце надходження запиту – Інтернет / локальна мережа, номер терміналу і т.д.).

Наступним аспектом, що визначає захищеність даних, є їх обсяг і застосовуються для зменшення обсягу механізми стиснення даних. По-перше, для різних даних повинні автоматично застосовуватися різні алгоритми стиснення і лише тоді, коли стискання дає вигоду. По-друге, адміністратор повинен мати можливість вибирати різні рівні стиснення, розуміючи, що за економію місця доведеться платити процесорним часом. По-третє, стиснення має бути прозоре для програм. При цьому СКБД повинна працювати зі стиснутими даними так само ефективно, як і з незстиснутими даними, не витрачаючи час на перебудову таблиць, на розтискання непотрібних блоків.

Традиційні алгоритми стиснення дозволяють стиснути дані в 2-3 рази. Деякі специфічні види стиснення дозволяють збільшити цю ступінь стиснення на порядок, але при цьому сильно сповільнюється виконання операцій DML з цими даними. Наприклад, відомо, що стиснення за полями збережених і заздалегідь відсортованих даних забезпечить дуже високий рівень стиснення (іноді в десятки разів). Проте спроби зміни таких таблиць в Oracle 11.2 показали, що час оновлення даних значно збільшується. Але для сховищ даних та історичних даних такий підхід цілком прийнятний.

Адміністратор повинен мати можливість для різних даних, різних таблиць і навіть частин таблиць вибирати ті варіанти стискання та зберігання (за строками, по стовпцях), які найбільш підходять з точки зору бізнес-використання цих даних. Причому, оскільки для додатків способи зберігання і стиснення прозорі, DBA може періодично міняти способи зберігання і стиснення. Цей підхід буде дуже корисний користувачам великих БД (наприклад, SAP-користувачам), які сьогодні змушені постійно збільшувати свої багатотерабайтні системи зберігання. Ну, і розумний механізм стиснення дозволяє залишати найсвіжіші та найбільш активно використовувані дані деякий час не стиснутими. Пізніше вони стискаються автоматично у фоновому режимі.

Ще один перспективний напрям розвитку – ізоляція DBA від даних.

Сьогодні адміністратор може бачити і змінювати всі дані в БД. Нові засоби захисту (такі як Oracle Data Vault option) дозволять DBA виконувати всі операції по адмініструванню БД, не дозволяючи йому бачити і змінювати дані. Також адміністратору можна буде обмежити набір дозволених до виконання операцій. Обмеження може бути прив'язане до імені DBA, часу, точці входу в систему.

1.3.2. Проблеми, викликані модифікацією структур даних

Як показано на рисунку 1.2 структура баз даних може істотно відрізнятись. Це значить, що перехід з однієї структури на іншу не може бути здійснений шляхом заміни операційної системи або сервера СКБД – такий перехід потребуватиме повного перепроєктування бази даних та переробки алгоритмів доступу до даних в програмному забезпеченні додатків.

Отже, фактично, перехід на нову структуру бази даних вимагає повної переробки системи. Причому, витрати на таку переробку можуть бути навіть більшими, ніж на первинну розробку, оскільки ця переробка має здійснюватись в умовах, коли в первинній системі вже були накопичені дані, а їх перенесення і забезпечення повноти та внутрішньої узгодженості може стати складною і відповідальною задачею.

1.3.3. Проблеми синхронізації даних в корпоративних сховищах даних

Проблеми синхронізації даних розглянемо на прикладі мережі біт-торент, яка покликана забезпечувати таку синхронізацію.

BitTorrent Sync синхронізує файли використовуючи однорангову мережу P2P, цей протокол може ефективно використовуватися для передачі великих файлів між безліччю пристроїв, цей принцип передачі даних використовується в таких додатках як Torrent, Transmission і BitTorrent. BitTorrent Sync відрізняється від хмарних сервісів тим, що користувацькі дані знаходяться на локальному носії, а синхронізація вимагає як мінімум одного пристрою підключеного до мережі.

BitTorrent Sync використовує шифрування даних за допомогою алгоритму AES з довжиною ключа в 128 біт, який може бути як створеним випадково так і вибраним користувачем. Ключ є похідною від секрету, який може бути поширений серед користувачів, які обмінюються даними. Дані надсилаються безпосередньо від пристрою до пристрою, якщо приймаючий пристрій недоступний (наприклад, перебуває за фаєрволом або виключено), то дані будуть спрямовані іншому вузлу мережі. Але дані будуть прийняті вузлом тільки в тому випадку, якщо йому відомий секретний ключ для розшифровки і перегляду даних. Вузол мережі може бути доданий до синхронізації, якщо йому буде передано відповідний секретний ключ, що дозволяє йому переглядати (RO-доступ) або змінювати (RW-доступ) дані.

Синхронізація робочих файлів між пристроями – найбільш очевидний спосіб використання, можна просто синхронізувати дані між декількома ПК або серверами, забезпечуючи наявність на кожному з пристроїв ідентичною інформації. Даний спосіб корисний в організації зберігання особистих і корпоративних даних.

Резервне копіювання – BitTorrent Sync може використовуватися в якості транспортного механізму між пристроєм, дані з якого вимагають резервування, і місця зберігання резервних копій. По суті, на резервується пристрої можна створити дзеркала потрібних директорій і реалізувати на їх базі інкрементне резервне копіювання або версійність.

Віддзеркалення і реплікація не самих критичних даних в хмарних або кластерних інтернет-додатках забезпечує зберігання ідентичного набору даних на декількох серверах, що дозволяє розподілити між ними навантаження. При цьому треба враховувати, що синхронізація потребує часу та обчислювальних ресурсів, тому в реальній роботі не варто спиратися на те, що всі вузли володіють актуальним і ідентичним набором даних. Також варто врахувати те, що репліки не усувають конфлікти інтелектуально – якщо в наборі даних є файли з ідентичними ім'ям і шляхом, то буде збережена тільки найбільш пізня

версія. Таким чином, якщо файли створюються програмним забезпеченням відразу на декількох вузлах (master-master реплікація), то використовують механізми іменування, які дозволяють уникнути конфліктів імен.

1.3.4. Проблеми, що визначаються предметною областю проектів

В якості прикладу, розглянемо декілька предметних областей застосування корпоративних інформаційних систем в проєкт сховища даних корпоративної інформаційної системи: SOHO та подібних.

В SOHO-проектах важливим з точки зору архітектури та організації системи також є вибір інструментального забезпечення інтеграції підсистем автоматизованої системи в єдину інформаційну систему. У відносно невеликих проектах така інтеграція, як правило, забезпечується безпосередньо на рівні програмного коду системи таким чином, що зв'язки між підсистемами визначаються в результаті безпосереднього звернення до керуючих об'єктам та об'єктам даних однієї підсистеми і викликом їх функцій обробки з інших підсистем. Наявність таких зв'язків спрощує розробку системи на початкових етапах реалізації, але, на жаль, не дозволяє простими засобами нарощувати систему надалі – у завершальних стадіях її розробки і на етапі експлуатації, коли будь-яка модифікація системи на рівні одного модуля неминуче призводить до зміни встановлених зв'язків між цим модулем та іншими модулями.

Для подолання цієї проблеми в складних системах реалізують більш гнучкі технології взаємодії підсистем шляхом реалізації керованих механізмів створення зв'язків між модулями, у тому числі – користувачем системи на етапі її експлуатації. Так, в системах, подібних MS Office або 1С, і в більшості інших складних систем реалізовані і надаються користувачам цих систем механізми, що дозволяють встановлювати нові правила ведення розрахунків і змінювати структуру бази даних шляхом простого створення нових зв'язків між модулями, модифікації існуючих або створення нових таблиць даних і т.д.. Реалізація

таких механізмів вимагає забезпечення специфічних технологій доступу до даних об'єктів, а в окремих випадках – ще й використання спеціалізованих мов програмування з підтримкою механізмів метапрограмування.

В проектах надання масового доступу до файлів. Реалізація і використання розподілених систем мають не лише плюси, але і мінуси, що пов'язані з особливостями забезпечення безпеки. Отримати контроль над розгалуженою і великою мережею P2P, або використати дірки в реалізації протоколів для власних потреб є бажаною метою для хакерів. До того ж захистити розподілену структуру складніше, ніж централізований сервер.

Величезну кількість ресурсів в мережах P2P, важко шифрувати/дешифрувати, тому більша частина інформації про IP-адреси і ресурси учасників зберігається і пересилається в незашифрованому вигляді, що робить її доступною для перехоплення. При перехопленні зловмисник не лише отримує власне інформацію, але і дізнається про вузли, на яких вона зберігається.

В клієнтах більшості крупних мереж ця проблема вирішується шляхом шифрування заголовків пакетів і ідентифікаційної інформації. З'являються клієнти з підтримкою технології SSL, впроваджуються спеціальні засоби захисту інформації про місце знаходження ресурсів.

Серйозною проблемою є поширення вірусів і підробка ID ресурсів з метою їх фальсифікації. Наприклад, в клієнті Kazaa використовується хеш-функція UUHash, яка дозволяє швидко знаходити ID для великих файлів навіть на малопотужних комп'ютерах, але при цьому залишається можливість для підробки файлів і запису зіпсованого файлу, що має такий же ID.

Щоб подолати таку проблему, клієнти повинні користуватися надійними хеш-функціями («деревами» хеш-функцій, якщо файл копіюється по частинах), такими, як SHA-1, Whirlpool, Tiger, і лише для вирішення некритичних завдань – контрольними сумами CRC. Для зменшення переданих об'ємів даних і полегшення їх шифрування можна застосувати компресію. Для захисту від

вірусів потрібно мати можливість зберігати ідентифікаційну метаінформацію про них, як це, наприклад, зроблено в мережі Gnutella2.

Іншою проблемою є можливість підробки ID серверів і вузлів. За відсутності механізму перевірки достовірності переданих службових повідомлень, наприклад за допомогою сертифікатів, існує можливість фальсифікації сервера або вузла (багатьох вузлів). Оскільки вузли обмінюються інформацією, підробка деяких з них приведе до компрометації всієї мережі або її частини. Закрите ПЗ клієнтів і серверів не є вирішенням проблеми, оскільки є можливість для реінжинірингу протоколів і програм (reverse engineering).

Виділені сервери і вузли періодично обмінюються між собою службовою інформацією і за необхідності додають підроблені сервери/вузли до чорного списку для блокування доступу.

Ще однією проблемою є нечесна поведінка частини клієнтів, які лише копіюють чужі файли, але нічого не пропонують іншим для копіювання (leechers).

Для боротьби з цим використовуються різні методи. В Bittorrent реалізовано схему «скільки блоків файлу отримав, стільки віддав». В eMule застосовано метод кредитів: скопіював файл – кредит зменшився, дозволив скопіювати свій файл – кредит збільшився (xMule – кредитна система із заохоченням поширювання рідкісних файлів).

1.3.5. Проблеми, що визначаються інструментальним забезпеченням

За окремими виключеннями будь який проект може з початкового рівня перерости в значний за обсягом. Таким чином, з точки зору предметної області проекти вимоги до проектів не можуть бути досить просто визначені відповідно до особливостей предметної області, але ці особливості напряду впливають на вибір інструментального забезпечення, що використовується при реалізації проекту. Отже, при цьому виникають істотні проблеми, пов'язані з специфікою стосовно предметної області та рівнем стосовно операційної системи задач, що

мають бути вирішені в проекті.

Для написання програм на низькому рівні абстракції, докладного і детального опису операцій, що враховують архітектура комп'ютера та операційної системи використовують низькорівневі мови програмування, такі як асемблер та С. У веб програмуванні асемблер не використовується, а мова С звичайно застосовується для розробки самого інструментального забезпечення, а не користувацьких додатків. В порівнянні з більш високорівневими мовами програмування програми мовою С звичайно вимагають більших витрат часу та високого рівня програмістів, отже збільшується і ймовірність появи помилок. Це можна вважати недоліком програмування на мовах низького рівня.

Мови програмування високого рівня дозволяють подолати недоліки низькорівневого програмування – використовувати більш розвинуті операції обробки агрегатних даних (списки, вектори, масиви, черги, стеки, сервіси операційної системи, комплексні числа тощо), не зважаючи на деталі їх реалізації на конкретній архітектурі. Програми при цьому виявляються більш короткими, надійними і універсальними, а процес їх складання скорочується. Програми на мовах високого рівня також легше читати та супроводжувати, в них простіше розібратися. Мови високого рівня бувають чотирьох видів: імперативні (процедурні), функціональні, логічні, об'єктно-орієнтовані. Основними об'єктами в імперативних мовах є змінні, оператори присвоювання, стандартні алгоритмічні конструкції.

Об'єктно-орієнтовані мови спрощують програмування з використанням технології об'єктно-орієнтованого програмування (Python, Java тощо), що, з точки зору можливого документально-орієнтованого представлення інформації, є корисним.

Враховуючи вищезазначене визначимо, що однією з основних проблем вибору інструментального забезпечення для реалізації активних сховищ корпоративних інформаційних систем слід вважати специфіку задач кожного конкретного проекту з точки зору вимог особливостей методів обробки даних,

що визначаються бізнес-процесами та реалізуються інструментальними засобами розробки.

1.3.6. Проблеми обсягу даних корпоративних інформаційних систем

При зростанні обсягу БД вузьким місцем стають канали передачі даних між сервером БД і пристроями зберігання. Можливість збільшити кількість і пропускну здатність таких каналів, а також знизити обсяг переданих даних за рахунок виконання частини операцій з перегляду та вибірки даних на рівні осередків зберігання, дозволяє зняти проблему каналів передачі. Така зв'язка обладнання та спеціального ПЗ, звана машина баз даних, давно описана в літературі. Приклади – Netezza, Datallegro, Teradata.

1.4. Постановка задачі дослідження

На основі зазначених чотирьох критерії та прикладів їх застосування для класифікації проектів можна зробити висновок, що будь які проекти можуть бути класифіковані в системі з чотирьох критеріїв. Це значить, що за наявності двійкової оцінки за кожним з критерієм існує всього 16 типів проектів

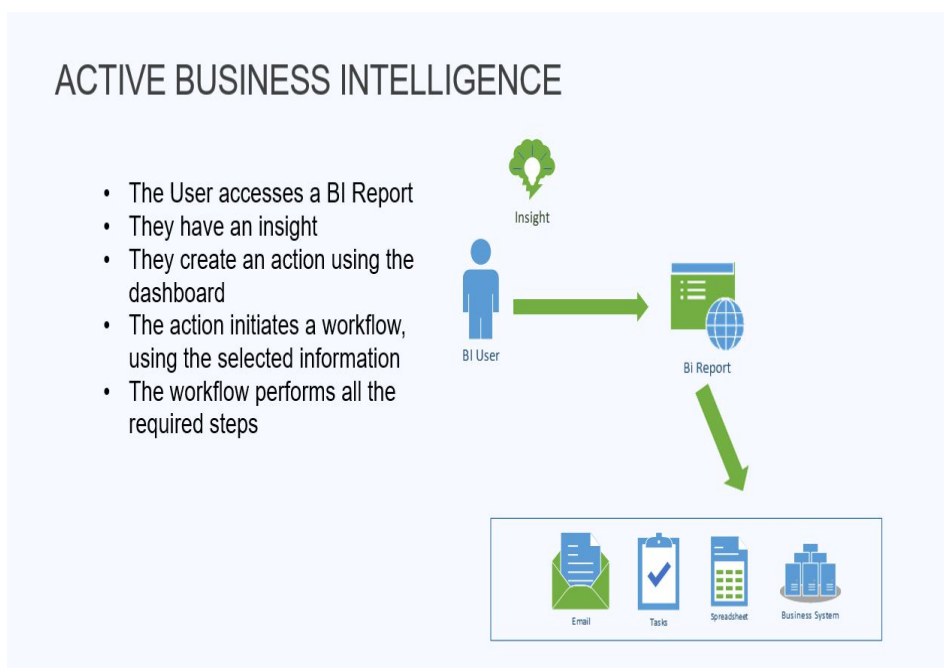


Рис. 1.9. Модель доступу до даних активного сховища даних АІС компанії

Виходячи з результатів аналізу виконаємо постановку задачі дослідження у наступному формулюванні: визначити можливості науково-обґрунтованого підходу до вибору технологій організації і доступу до даних, та можливостей їх застосування в проектах аналізу даних корпоративних інформаційних систем, що дозволять з наявного спектру інструментального забезпечення, методів та засобів збереження, транспортування, захисту та обробки даних виконувати розробку корпоративних інформаційних систем та, зокрема, проєктів сховищ даних корпоративних інформаційних систем з найкращими результатами.

1.5. Висновки до розділу 1

В розділі були розглянуті технології організації і доступу до даних у корпоративних інформаційних системах та, зокрема, проєкт сховища даних корпоративної інформаційної системи, що визначаються інструментальним забезпеченням, методами та засобами збереження, транспортування, захисту та обробки даних корпоративних інформаційних систем та проєктів сховищ даних корпоративних інформаційних систем.

Визначено, що широке представлення та популярність технологій збереження даних у файлових сховищах та інтеграція корпоративних систем у загальний інформаційний простір вказують на те, що при збереженні значних обсягів інформації корпоративні системи збереження даних можуть використовувати такі сховища та надавані ними інструменти доступу до даних і аналізу даних як одну з технологій збереження, доступу і обробки даних.

Визначено, що широкий спектр задач корпоративних інформаційних систем призводить до накопичення значних обсягів даних, окремі з яких використовуються постійно, а інші лише зрідка та при вирішенні вузького кола задач; сумісне збереження таких даних призводить до зменшення ефективності сховища даних. Як наслідок, все це призводить до необхідності здійснення безперервної модифікації структури даних у сховищі даних через зміни у

діяльності компанії, законодавстві тощо; модифікація структури даних ускладнює оцінку ефективності сховища даних.

Виконано постановку задачі дослідження.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ ТА МЕХАНІЗМІВ ОЦІНКИ ЕФЕКТИВНОСТІ СХОВИЩ ДАНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1. Визначення критеріїв оцінки методів та засобів розробки проектів сховищ даних корпоративних інформаційних систем

2.1.1. Визначення критеріїв оцінки методів та засобів збереження і аналізу даних в проектах корпоративних інформаційних систем

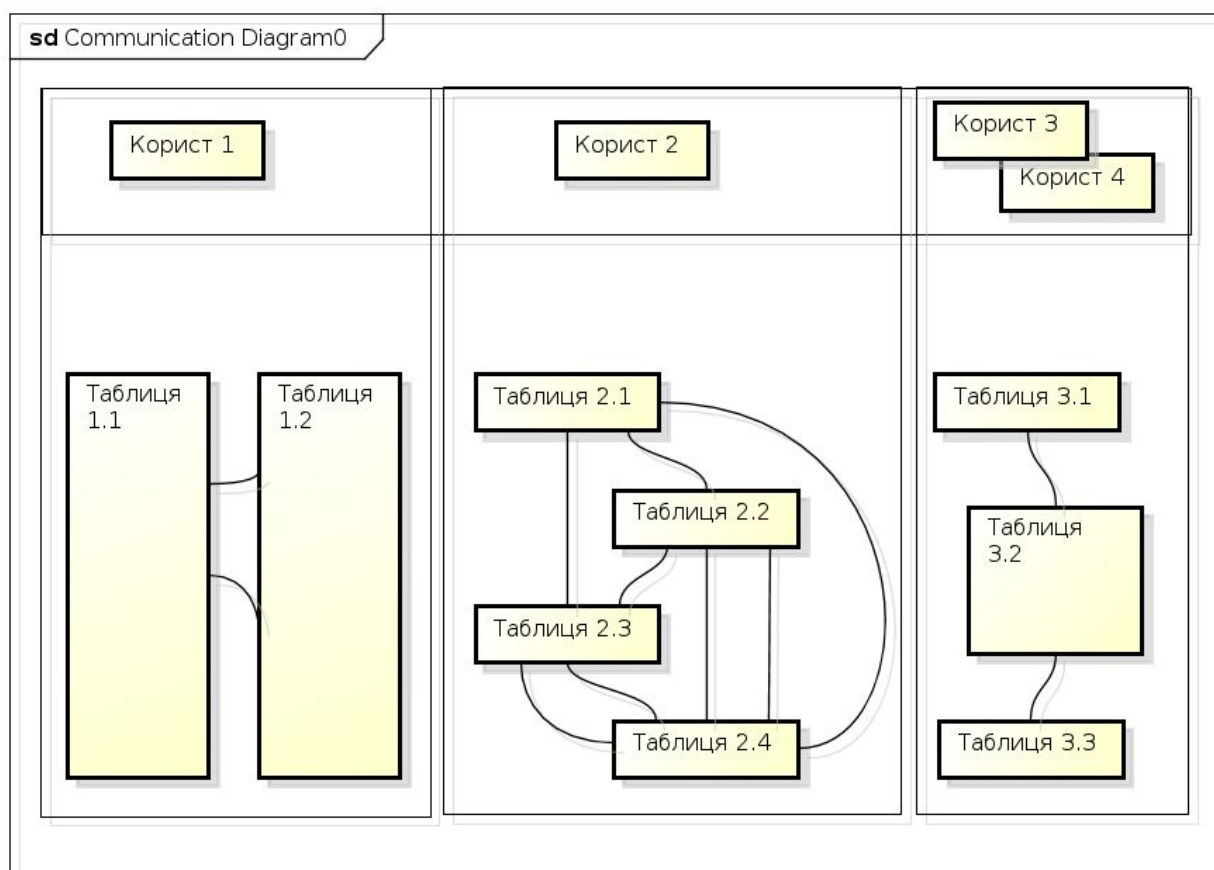
Враховуючи проблеми, пов'язані з бізнес-логікою і обробкою і аналізом даних, обсягом даних та обсягом запитів до СКБД, активні сховища даних можуть розглядатись як навантажені СКБД. Такі СКБД можуть розглядатись як:

- системи з великими обсягами даних;
- системи зі складною організацією даних;
- системи з великим обсягом запитів (як правило — зі значним числом користувачів).

Візуально різновиди навантажених СКБД можуть бути представлені у вигляді моделі (рис 2.1).

Як було показано вище, хоча основною метою СКБД є управління наборами даних, які за визначенням мають дуже великий обсяг і неминуче у багато разів перевищують обсяг оперативної пам'яті, проведені дослідження показали, що доступ до даних в загальному випадку підкоряється правилу “80/20” або “90/10” (відомому як закон Парето): 80-90% всіх звернень виконуються для отримання 10-20% даних. Більше того, виявилось, що до постійно потрібних даних, звернення до яких становлять 80-90% від усіх звернень, знову застосовано правило 80/20” або “90/10”. Таким чином, приблизно 80-90% усіх звернень до даних пов'язані з доступом до приблизно 1% агрегованих даних. Слід зазначити, що це відношення включає звернення до службових даних СКБД (що включають індекси В-дерев і т. п.), отримання і обробка яких зазвичай приховані від прикладного програміста. Хоча бажано мати коефіцієнт попадань в кеш приблизно на рівні 95%, з економічних

міркувань неможливо сліпо забезпечити обсяг кешу в пам'яті, що дозволяє розмістити 10-20% всіх даних: навіть для скромних баз даних об'ємом 5 Гбайт це зажадало б збільшення розміру основної пам'яті до 500 Мбайт. Проте зазвичай завжди можливо забезпечити кеш обсягом в 1% всіх даних навіть для дуже великих баз даних. Максимальний обсяг основної пам'яті сучасних серверів може бути дуже великим. Слід зазначити, що попередні оцінки є доволі "середніми" і в реальних задачах можуть значно відрізнятись як в більшу, так і в меншу сторону. Отже, аналіз ефективності сховища даних і інформаційної системи в цілому слід здійснювати з обов'язковим врахуванням обсягів доступної оперативної пам'яті і впливу обсягів пам'яті на ефективність.



powered by Astah

Рис. 2.1. Різновиди навантажених СКБД: з великим обсягом даних (ліворуч), зі складною структурою (у центрі), зі значним числом користувачів (праворуч)

На основі результатів аналізу сучасних систем управління базами даних, їх переваг та недоліків можна визначити декілька критеріїв щодо можливостей застосування СКБД кожного з типів для вирішення задач збереження даних корпоративних інформаційних систем. Ці критерії наступні:

- Відповідності — сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню проекту, наприклад, в комерційних проектах повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних тощо;
- Критерій розвитку активного сховища даних (в даному випадку стабільні краще);
- Критерій підтримки СКБД інструментальними засобами (бажана підтримка сучасних мов програмування);
- Критерій вартості входження та супроводу СКБД активного сховища даних (прості і безкоштовні краще у зв'язку з коротким часом життя проектів);
- Критерій підтримки інструментів з сучасними технологіями програмування (кращі ті інструменти, що відносно простими засобами дозволяють реалізувати більш складну бізнес логіку, наприклад з підтримкою технологій узагальненого програмування або метапрограмування);
- Критерій простоти реорганізації модифікації структури бази даних (так, щоб кожен окремий запит звертався тільки до тих таблиць, які містять дані, що відносяться до цього запиту. Як наслідок великі таблиці активних сховищ даних повинні бути розбиті на кілька, може однотипних, але маленьких).

2.1.2. Визначення критеріїв оцінки методів та засобів розробки проектів активних сховищ даних корпоративних інформаційних систем

Враховуючи, що однією з основних проблем вибору інструментального забезпечення для реалізації проектів сховищ даних корпоративних інформаційних систем слід вважати специфіку задач кожного конкретного проекту з точки зору вимог особливостей методів обробки даних, що визначаються бізнес-процесами і використовують дані безпосередньо з запитів (транзакційний пакет даних, рисунок 2.1). Зважаючи на вищенаведене, визначимо наступні ключові критерії щодо застосування інструментальних засобів активних сховищ даних в корпоративних і проєкт сховища даних корпоративної інформаційної системи (Таблиця 1.3).

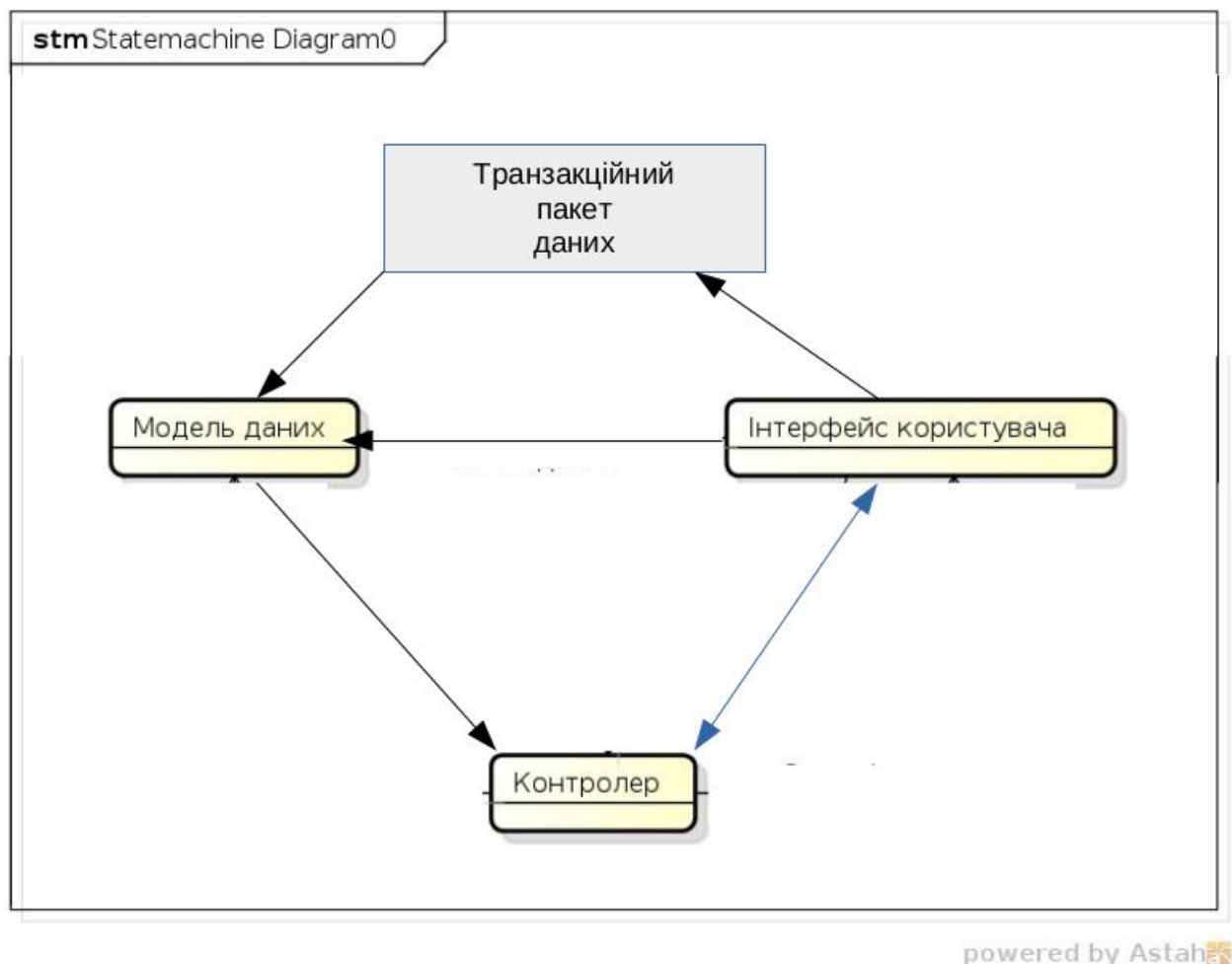


Рис. 2.2. Базова MVC-модель додатка з пасивним сховищем даних

Активне сховище даних обумовлює високе навантаження на сервер баз

даних в частині отримання великих масивів даних з безперервним поповненням бази даних, що визначає високі вимоги до потужності і надійності СКБД.

Активне сховище даних обумовлює високе навантаження на сервер додатків в частині безперервної обробки великих масивів даних для вирішення задач бізнес-аналітики.

Обробка даних повинна відбуватись в режимі ізоляції транзакцій для забезпечення відповідності результатів аналізу даних поточному запиту і наявним даним у базах даних.

Залежно від вимог щодо забезпечення кожної з цих областей наявні корпоративні та проєкт сховища даних корпоративної інформаційної системи можуть бути поділені на категорії з різними вимогами. Нижче наведені приклади (таблиця 2.1.).

Таблиця 2.1. Ключові критерії щодо застосування інструментальних засобів в корпоративних та проєкт сховища даних корпоративної інформаційної системи

№	Критерій	Примітка
1	Збереження та надання інформаційних ресурсів	
2	Комунікації та обмін інформацією	
3	Масові обчислення	
4	Управління соціальними та технічними системами	

Наведений перелік звичайно не є повним, але він дає можливість виконати оцінку адекватності зазначених критеріїв стосовно предметних областей зберігання і аналізу даних. Як наслідок, для забезпечення цих критеріїв можуть застосовуватись різні інструментальні засоби, кожен з яких

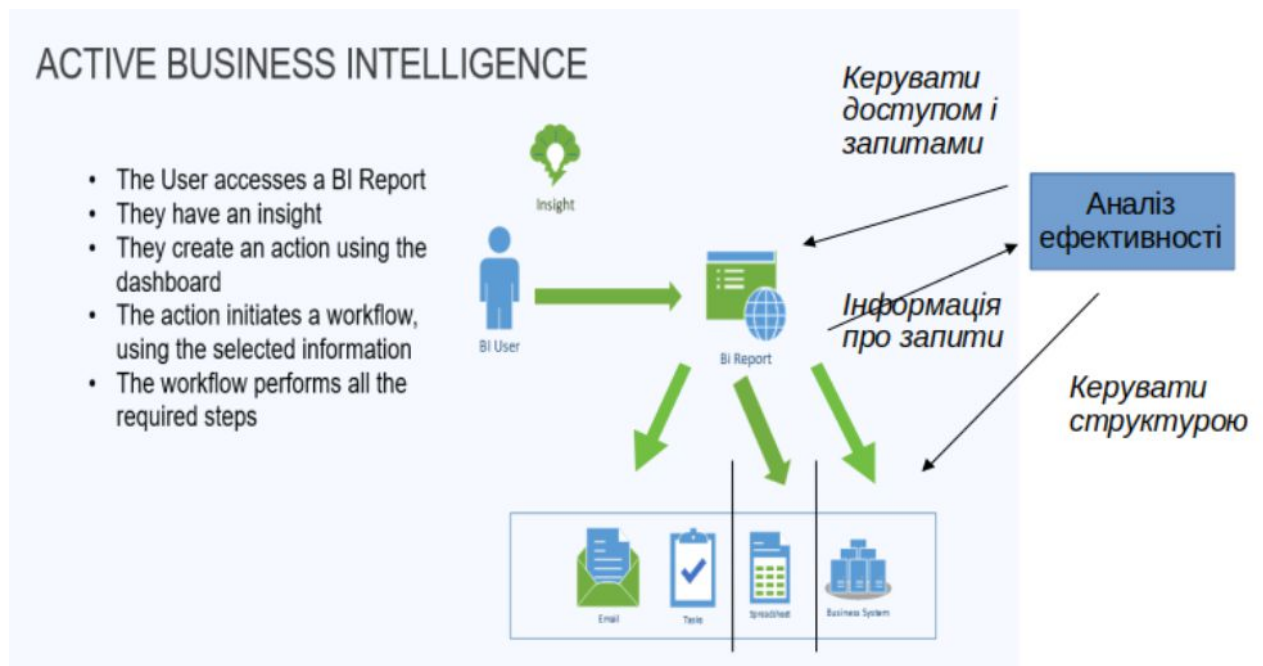
тим чи іншим чином відповідає вимогам щодо критеріїв.

Очевидно, що для забезпечення відповідності властивостей систем зазначеним вище критеріям якості інформаційних систем знадобляться і відповідні інструментальні засоби, методи проектування, механізми збереження, обробки та передачі даних.

2.2. Розробка моделей оцінки якості механізмів збереження та доступу до активних сховищ даних

2.2.1. Критерії оцінки якості технологій збереження та доступу до активних сховищ даних корпоративних інформаційних систем

Враховуючи визначені в попередньому розділі (пп. 1.4) критерії, що визначають якість механізмів збереження та доступу до даних, та особливості MVC-моделі додатків корпоративних систем, визначимо додаток з активним сховищем даних як такий, що використовує при обробці даних поточного запиту не лише дані, що зберігаються у сховищі даних, але і дані із самого



запиту (рисунок 2.2).

Рис. 2.2. Базова модель аналізу ефективності сховища даних в системі з

активним сховищем даних

На основі зазначеного визначимо підходи до розробки моделей оцінки якості механізмів збереження та доступу до даних.

Дамо розгорнуте трактування різновидів навантажених СКБД корпоративних інформаційних систем (рисунок 2.3) та окремих із зазначених критеріїв якості проектів активних сховищ даних.

Критерій відповідності полягає в тому, що сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню корпоративного або проєкт сховища даних корпоративної інформаційної системи, наприклад, в комерційних проєктах повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних тощо.

Таблиця 2.2. Приклади ранжування проектів корпоративних систем за категоріями вимог

№	Предметна область	Значення для критерію			
		Збереження та надання ресурсів	Комунікації, захист та обмін інформацією	Масові обчислення	Управління іншими системами
1	Електронні ЗМІ	високе	низьке	низьке	низьке
2	Електронна торгівля	високе	середнє	низьке	низьке
3	Навчання	середнє	середнє	середнє	середнє
4	Інженерні розрахунки	середнє	середнє	високе	низьке
5	Соціальні мережі	низьке	високе	низьке	низьке

6	Електронний уряд	високе	середнє	низьке	високе
7	Системи моніторингу та управління	високе	високе	високе	високе

Критерій підтримки проекту інструментальними засобами полягає в тому, що окрім поточної розробки будь яку систему необхідно підтримувати і надалі. При цьому непопулярність інструментальних засобів в майбутньому може призвести до проблем з пошуком та залученням відповідних фахівців. Як наслідок, в проекті бажана орієнтація на сучасні мови веб-програмування і опису даних.

Критерій простоти і вартості СКБД полягає в тому, що прості і безкоштовні СКБД є кращим вибором з погляду на відносно короткий термін життя окремих проєктів сховищ даних корпоративних інформаційних систем.

Критерій простоти реорганізації бази даних полягає в тому, щоб кожен окремий запит звертався тільки до тих таблиць, які містять дані, що відносяться до цього запиту. Як наслідок великі таблиці повинні бути розбиті на кілька, може однотипних, але маленьких. Приклад: інтернет форум, в якому повідомлення форуму краще зберігати не в одній великій таблиці по всіх темах форуму, а в декількох невеликих таблицях, але містять повідомлення тільки однієї теми.

Критерій простоти модифікації структури БД визначається тим, що у зв'язку з високим навантаженням і коротким часом життя інтернет проєктів структура баз даних часто зазнає змін через відносно спрощені технології проектування ПЗ корпоративних та проєктів сховищ даних корпоративних інформаційних систем.

Визначимо, що оцінка якості проєктів має враховувати два основні аспекти:

- якість використання програмної системи, тобто система має задовольняти експлуатаційним вимогам;
- якість розробки програмної системи, тобто система має задовольняти технологічним вимогам.

Виходячи з того, що система, яка не може використовуватись, має нульову якість так само як і система, що не може бути реалізована, визначаємо, що ці аспекти можуть бути враховані в загальній оцінці якості як середнє геометричне за кожним з аспектів:

$$c = \frac{c_u - c_d}{c_u + c_d} \rightarrow \max, \quad (2.1)$$

де: c – оцінка, c_u – відповідність вимогам бізнес аналітики; c_d – відповідність вимогам збереження; c_a – відповідність технологіям.

В свою чергу, відповідність вимогам бізнес аналітики визначається відповідністю механізмів доступу до даних та механізмів збереження даних:

$$c_u = 1/n \sum_{i=1}^n f(m_i, t_i, m_i, s_i), \quad (2.2)$$

де $f(t_i, s_i)$ – функція обчислення критерію якості окремої транзакції залежно від обсягу даних і часу обробки.

$$c_d = 1/n \sum_{i=1}^n g(m_i, t_i, m_i, s_i), \quad (2.3)$$

де $g(t_i, s_i)$ – функція обчислення критерію якості окремої транзакції залежно від обсягу даних і часу доступу до даних.

$$\sum_{i=1}^n m_i = 1.0,$$

де m_i – відносна ймовірність припустимого варіанта конфігурації даних СКБД і транзакційного пакету.

Зважаючи на результати аналізу, визначаємо, що відповідність вимогам розробника забезпечується відповідністю механізмів збереження та доступу до даних залежно від предметної області задачі, а відповідність вимогам розробника – відповідністю інструментальних засобів (в тому числі мови

програмування) щодо можливостей забезпечення збереження та доступу до даних.

2.2.2. Механізми забезпечення якості активних сховищ даних корпоративних інформаційних систем

Відповідно до визначених вище критеріїв якості, згідно до таблиці 2.1 та (2.2) визначимо значення критерію c_s як сумму:

$$c_a = \frac{k_e \cdot c_e + k_q \cdot c_q + k_v \cdot c_v + k_w \cdot c_w + k_r \cdot c_r}{k_e + k_q + k_v + k_w + k_r}, \quad (2.4)$$

де: e — захищеність; q — швидкість розробки; v — наочність коду; w — швидкість виконання базових операцій; r — простота тестування.

Коефіцієнти k, c будуть визначатись на основі результатів експертного оцінювання.

Серед основних механізмів забезпечення якості в активних системах збереження даних найбільш цікавим, з точки зору корпоративних технологій, є масштабування баз даних — партиціонування, реплікація і шардинг.

СКБД — це дуже часто «вузьке місце» в продуктивності інтернет-додатків. У момент, коли сервер баз даних не може впоратись з навантаженнями, проводиться масштабування. Розглянемо основні способи збільшення продуктивності СКБД.

Розглянуті механізми масштабування застосовні як для реляційних баз даних, так і для NoSQL-сховищ. Зрозуміло, що у всіх баз даних і сховищ є своя специфіка, тому ми розглянемо тільки основні напрямки і в деталі реалізації вдаватися не будемо.

Партиціонування (partitioning) — це розбиття таблиць, що містять велику кількість записів, на логічні частини за деякими обраним адміністратором критеріям. Партиціонування таблиць ділить весь обсяг операцій з обробки даних на кілька незалежних і паралельно виконуються потоків, що істотно прискорює роботу СКБД. Для правильного конфігурування параметрів

партіціонування необхідно, щоб в кожному потоці було приблизно однакову кількість записів.

Наприклад, на новинних сайтах має сенс партіціонувати записи за датою публікації, так як свіжі новини на кілька порядків більш затребувані і частіше потрібна робота саме з ними, а не з усіх архівом за роки існування новинного ресурсу.

Реплікація (replication) – це синхронне або асинхронне копіювання даних між декількома серверами. Провідні сервера називають майстрами (master), а ведені сервера – слейвами (slave). Майстри використовуються для зміни даних, а слейви – для зчитування. У класичній схемі реплікації зазвичай жоден майстер і кілька слейвів, оскільки в більшій частині корпоративних та проєктів сховищ даних корпоративних інформаційних систем операцій читання на кілька порядків більше, ніж операцій запису. Однак у більш складній схемі реплікації може бути і кілька майстрів.

Наприклад, створення декількох додаткових slave-серверів дозволяє зняти з основного сервера навантаження і підвищити загальну продуктивність системи, а також можна організувати слейви під конкретні ресурсомісткі завдання і таким чином спростити складання серйозних аналітичних звітів – використовуваний для цих цілей slave може бути навантажений на 100%, але на роботу інших користувачів додатку це не вплине.

Шардінг (sharding) – це прийом, який дозволяє розподіляти дані між різними фізичними серверами. Процес шардінга припускає рознесення даних між окремими Шарден на основі якогось ключа шардінга. Пов'язані однаковим значенням ключа шардінга суті групуються в набір даних по заданому ключу, а цей набір зберігається в межах однієї фізичної Шарда. Це істотно полегшує обробку даних. Наприклад, в системах типу соціальних мереж ключем для шардінга може бути ID користувача, таким чином всі дані користувача будуть зберігатися та оброблятися на одному сервері, а не збиратися по частинах з декількох джерел.

Крім цього, істотним з точки зору задоволення критеріям оцінки якості, є механізми зберігання та індексування даних в БД.

Серед популярних механізмів зберігання даних найбільш цікавим для реалізації вузькоспеціалізованих і високонавантажених серверів БД, до яких відносяться БД корпоративних та проєктів сховищ даних корпоративних інформаційних систем, представляються In-memory СКБД реального часу як кеш для комерційних СКБД.

Практично всі універсальні комерційні СКБД сьогодні – це СКБД, орієнтовані на роботу з дисками. Така архітектура, навіть при дуже гарній налаштування додатки, може дати час відгуку порядку декількох мілісекунд. Однак для багатьох додатків (наприклад, для систем, що працюють в реальному часі) потрібно швидкодію на порядок вище (час відгуку – мікросекунди). Для отримання такої швидкодії використовуються спеціальні СКБД, що працюють в оперативній пам'яті (In memory DB) і мають спеціальну архітектуру, спеціальні способи зберігання, адресації і індексації даних, спеціальні механізми оптимізації і буферизації і т.д. Дані на диск вони не скидають або роблять це у фоновому асинхронному режимі. Надійність роботи та обробка збоїв реалізується спеціальними методами (наприклад, за рахунок реплікації пам'ять – пам'ять). Багато хто не дуже важливі функції таких СКБД можна відключати (наприклад, журналювання) на догоду швидкодії.

2.2.3. Механізми забезпечення якості доступу до активних сховищ даних корпоративних інформаційних систем

Відповідно до визначених вище критеріїв якості, згідно до пункту 1.4.2 та (2.2) визначимо значення критерію c_a як сумму:

$$c_{us} = k_b \cdot c_{ub} + k_p \cdot c_{up}, \quad (2.5)$$

де: $k_b \cdot c_{ub}$ – ваговий коефіцієнт та значення критерію безпеки передачі даних;
 $k_p \cdot c_{up}$ – ваговий коефіцієнт та значення критерію доступності протоколів передачі даних.

З точки зору оцінки якості доступу до даних в складних корпоративних системах перевагу слід віддавати тим рішенням, які, насамперед, дозволяють реалізувати необхідну бізнес логіку (так як в іншому випадку вже не важливі такі характеристики, як швидкість, захищеність і т.д.). Виходячи зі складності таких систем, для їх розробки застосовують, як правило, об'єктно-орієнтовані технології програмування та проектування. При цьому сховища даних також повинні бути об'єктними. Враховуючи, що реляційні СКБД все ще популярні і широко використовуються рішенням задачі об'єктного доступу до даних в реляційному сховищі є ORM-системи.

Для забезпечення якості програміста ORM-система повинна виглядати як постійне сховище об'єктів. На практиці все ORM-системи, як правило, виявляють себе саме таким чином, забезпечуючи можливість ігнорування реляційної природи бази даних як набору таблиць і відносин між ними.

Крім цього, враховуючи, що програмне забезпечення для складних систем зазвичай розробляється відповідно до принципів об'єктно-орієнтованого програмування, необхідність використання ORM-систем так само виникає як результат невідповідності способів організації структурованих даних систем управління реляційними базами даних способам організації об'єктів як структурованих даних і механізмів їх обробки в програмах, які використовують такі бази даних. Ця невідповідність полягає в тому, що об'єкти програми можуть перебувати у відносинах агрегації, асоціації та використання, а реляційні бази даних підтримують тільки відносини асоціації, причому, відносини асоціації "багато-до-багатьох" більшістю реляційних баз даних взагалі не підтримується.

Захист клієнта від можливості отримання недостовірних даних або втрати власних базуються на наступних механізмах:

- захисті каналів передачі даних та захисті від потрапляння на клієнтську машину недостовірних даних, що має місце при підміні сервера, коли за тією самою адресою працює інший, недостовірний сервер (який може

- видавати себе за достовірний шляхом надання очікуваного контенту) або в результаті автоматичного перенаправлення запиту;
- впровадження в сторінку шкідливого коду, який буде виконаний на комп'ютері користувача при відкритті ним цієї сторінки (XSS-атака).

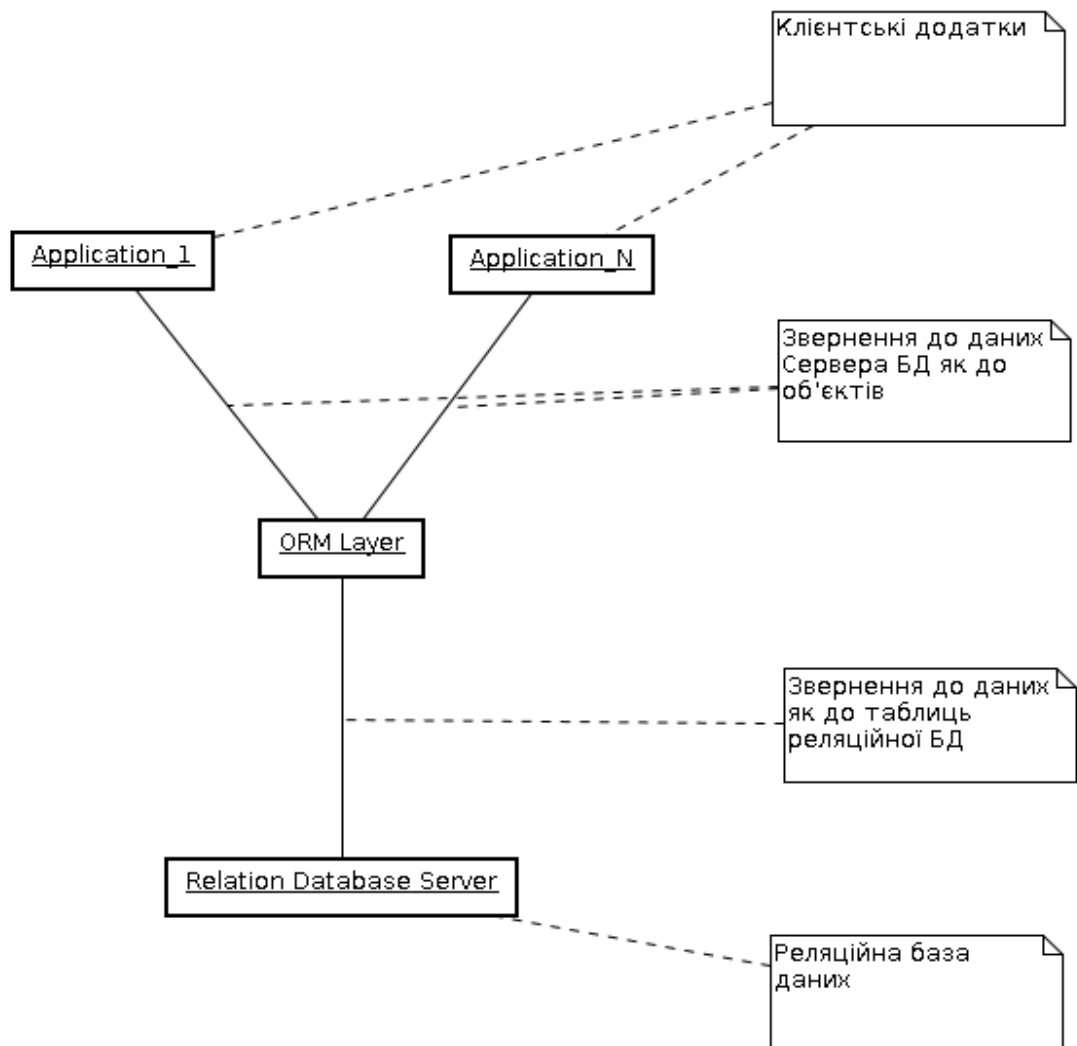


Рис. 2.4. Компонентна модель доступу до реляційної бази даних з Object Relational Mapping (ORM)

Найбільш часто експлуатуються такі уразливості на сайтах, де користувачі можуть публікувати свій контент, роблячи його доступним для інших. Зловмисник, при відсутності належної фільтрації, може або безпосередньо використовувати уразливість, або заражати системи

користувачів. Розробники веб проєктів для запобігання атак цього типу змушені або просто обмежувати можливості користувачів публікувати на сайті що-небудь, або ретельно фільтрувати надходять від користувачів дані.

2.2.4. Механізми доступу та збереження даних клієнта активних сховищ даних корпоративних інформаційних систем

Браузерні або клієнтське кешування. Кешування статичних ресурсів (картинок, скриптів, стилів) і незмінних сторінок на стороні браузера може заощадити час завантаження сторінок, якщо користувач відвідує сайт багаторазово або при відвідуванні переглядає кілька сторінок, які використовують однакові ресурси.

Найчастіше на стороні браузера кешуються файли зображень, JS і CSS файли. Трохи рідше має сенс кешувати сторінки і бінарні файли (медіа-файли, PDF, завантажувані документи, архіви і т. д.).

Статичні ресурси повинні мати хоча б тижнєве час життя кешу, а краще кешувати їх відразу на місяць – з врахуванням сезонного характеру окремої інформації. Таким чином, ці ресурси будуть тільки один раз отримують з сервера, а потім браузер буде або відразу використовувати локальну копію (якщо вказано заголовок Expires або Cache-Control), або після перевірки на незмінність (якщо вказано заголовок Last-Modified або Etag).

Заголовки Expires і Cache-Control: max-age. Як значення у цих заголовків використовується дата. До тих пір, поки вона не настала, браузер буде без будь-яких додаткових перевірок використовувати закешовану версію ресурсу. Expires підтримується трохи ширше, ніж Cache-Control: max-age, тому краще використовувати саме його.

Заголовки Expires і Cache-Control: max-age. Як значення у цих заголовків використовується дата. До тих пір, поки вона не настала, браузер буде без будь-яких додаткових перевірок використовувати закешовану версію ресурсу. Expires підтримується трохи ширше, ніж Cache-Control: max-age, тому краще

використовувати саме його.

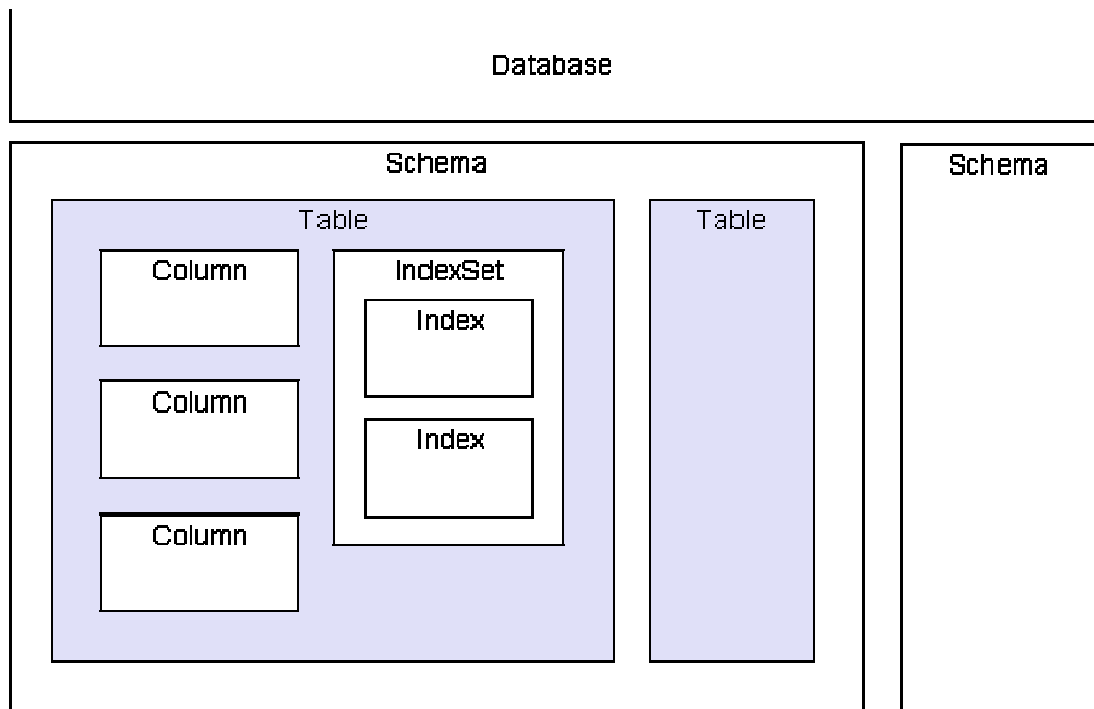


Рис. 2.5. Логічна структура даних в ORM-системі

Заголовки Last-Modified і Etag. Як значення заголовка Last-Modified використовується дата, а для заголовка ETag – довільна рядок. Якщо використовуються ці заголовки, то браузер, перш ніж використовувати закешований ресурс, отримає з сервера поточне значення заголовка і порівняє його з заголовком закешованої версії – якщо дані співпадуть, то буде використовуватися локальна версія, а якщо ні – відбудеться повторне завантаження. Запити перевірки відбуваються швидше, ніж повне завантаження ресурсів, що дає приріст продуктивності сайту при повторному використанні ресурсів.

Оптимальна стратегія клієнтського кешування. Для різко мінливих файлів варто поставити Expires на дату, яка настане через місяць, а в момент зміни цих ресурсів використовувати метод «відбитків пальців» (fingerprinting) – додавати до імені файлу невелику частину хешу, яка утворюється на основі його змісту. Таким чином, вся статика завжди буде закешована на клієнті, а при зміні цієї

самої статички буду змінені імена файлів і вони автоматично будуть перезавантажуватися браузером.

Таблиця 2.3. Типи та періодичність кешування ресурсів інтернет-сторінок на стороні клієнта

<i>№</i>	<i>Типи ресурсів</i>	<i>Призначення</i>	<i>Період</i>
1	Файли зображень	Медійна інформація	1 місяць
2	Скрипти JS	Анімація, обробка даних на стороні клієнта	1 місяць
3	CSS файли	Стили представлення, оформлення	1 місяць
4	Статичні сторінки	Інформація, що змінюється розробником контенту	1 день – 1 тиждень
5	Бінарні файли	Інформація, що змінюється сервером за запитом	до 1 дня

Власне техніка fingerprinting активно використовується для статичних ресурсів (стилів і скриптів) у багатьох фреймворках (зокрема, в Ruby on Rails). А масову проставку для всіх типів статичних файлів заголовка Expires можна реалізувати на стороні інтернет-сервера (в nginx, наприклад).

Таблиця 2.4. Елементи сторінки, що впливають на періодичність кешування ресурсів інтернет-сторінок на стороні клієнта

<i>№</i>	<i>Типи заголовку</i>	<i>Тип</i>	<i>Примітки</i>
----------	-----------------------	------------	-----------------

1	Expires	дата	
2	Cache-Control: max-age	дата	Обмежена підтримка
3	Last-Modified	дата	
4	Etag	рядок	

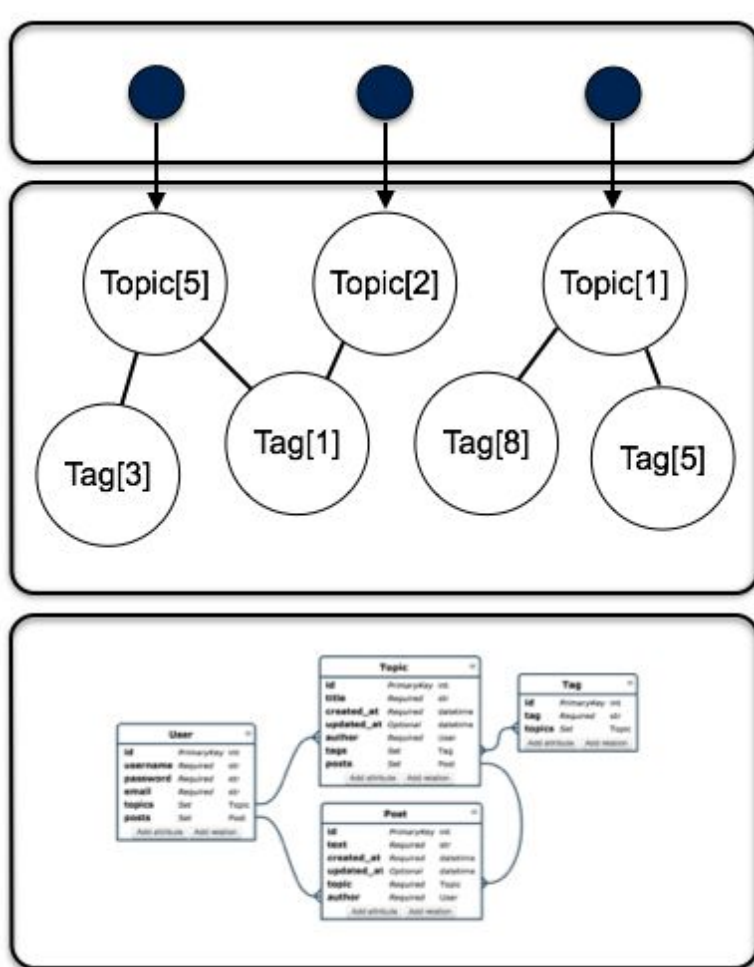


Рис. 2.6. Фізична структура даних в ORM-системі (зверху-донизу: рівень доступу; рівень організації, рівень збереження)

2.3. Розробка моделей вибору методів та засобів збереження та доступу і аналізу даних в проектах корпоративних інформаційних систем

Вище були визначені критерії оцінки методів та засобів збереження інформації, які, при бінарному значенні («потрібні», «непотрібні») дозволяють

визначити вимоги до будь якого сайту з 16 (або 2^{**4}) можливих типів сайтів. Очевидно, що необхідні механізми визначення наслідків порушення цих критеріїв для визначення напрямків та заходів, що мають бути застосовані при покращенні роботи кожного сайту.

Найпростіший спосіб введення таких механізмів полягає у використанні системи вагових коефіцієнтів з можливістю приведення оцінок за критеріями до скаляра для подальшого співставлення з інструментами, які дозволять найкращим чином досягти визначених критеріїв.

Очевидно, що для кожної з предметних областей значення цих вагових коефіцієнтів будуть відрізнятись (що було показано в п. 1.4.2). В якості прикладу розглянемо проекти з двох предметних областей (табл. 2.3).

Таблиця 2.5. Основні параметри активних сховищ корпоративних ІС

Критерій	Навчання	Соціальні мережі
Збереження та надання інформаційних ресурсів	0.5	0.2
Комунікації та обмін інформацією	0.6	0.8
Масові обчислення	0.4	0.2
Управління соціальними та технічними системами	0.4	0.2

Також визначимо оцінки для інструментальних засобів розробки корпоративних систем і сховищ даних різними мовами програмування щодо можливостей забезпечення зазначених вище критеріїв (табл. 2.4, для прикладу для мов C/C++ та Python).

Також визначимо якості кожного з інструментальних засобів розробки стосовно розробників системи (табл. 2.5, для прикладу для мов C/C++ та Python).

Таблиця 2.6. Основні параметри інструментальних засобів щодо забезпечення властивостей системи

Критерій	C/C++	Python
Збереження та надання інформаційних ресурсів	0.5	0.5
Комунікації та обмін інформацією	0.6	0.6
Масові обчислення	0.9	0.2
Управління соціальними та технічними системами	0.4	0.2

Таблиця 2.7. Основні параметри інструментальних засобів щодо складності розробки

Критерій	C/C++	Python
Захищеність програмного забезпечення	0.9	0.1
Швидкість розробки ПЗ проекту	0.3	0.8
Наочність коду	0.4	0.9
Швидкість обробки запитів	0.8	0.5
Валідність коду	0.6	0.7

Враховуючи наведене вище визначимо основну ідею проєкту. Ідея проєкту полягає у такому розподілі даних у сховищі, щоб окремі частини сховища максимально відповідали структурам запитів кожної окремої бізнес-задачі за умови прийнятної складності сховища та часу доступу до даних.

Для втілення зазначеної ідеї необхідно обчислювати “потужність” зв’язків між окремими таблицями у запитах. Для цього пропонується створити додатковий набір даних, в який будуть заноситись лічильники використання окремих таблиць у кожному окремому запиті (Табл. 2.8).

Таблиця 2.8. Приклад матриці лічильника звернення до таблиць БД за запитами

Запит	Таблиця 1	Таблиця 2	Таблиця 3	...	...	Таблиця N
Запит 1	+	-	+			
Запит 2	+	-	+			
....						
Запит M	-	+	+			

За результатами, занесеними у цей набір даних (Табл. 2.8) може бути створений набір, що буде визначати “потужності” зв’язків між таблицями (Табл. 2.9).

Таблиця 2.9. Приклад матриці залежностей між таблицями БД за бізнес задачами

	Таблиця 1	Таблиця 2	Таблиця 3	...	...	Таблиця N
Таблиця 1	-	0	2			1
Таблиця 2	0	-		1		1
Таблиця 3	2	1	-			1
...				-		
...					-	
Таблиця N	1	1	1			-

2.4. Вибір інструментального забезпечення розробки проектів аналізу ефективності сховищ даних корпоративних інформаційних систем

Враховуючи результати аналізу, наведені в розділі 1, можна стверджувати,

що розробку проєктів бізнес аналізу даних в корпоративних інформаційних системах та проєкт сховища даних корпоративної інформаційної системи на поточний момент доцільно виконувати на одній з наступних мов: PHP, Python, Perl, Ruby та Java. Кожна з цих мов має розвинуті інструментальні засоби (фреймворки), написані цими ж мовами. Хоча кожна з зазначених мов програмування має свої особливості, їх спільні переваги з погляду технологій програмування корпоративних інформаційних систем та проєктів сховищ даних корпоративних інформаційних систем наступні:

- Висока швидкість розробки. Це обумовлено як технічними особливостями фреймворку (наприклад, продумані угоди спрощують конфігурацію), так і інструментами розробки (консольні утиліти і генератори, готові бібліотеки, розширення і модулі).
- Простота реалізації складної бізнес-логіки — конвенції написання програмного коду на базі цих мов дозволяють створювати зрозумілий програмний код, який згодом може бути без проблем модифікований в адекватні терміни, а дотримання цих конвенцій робить програмний код супроводжуваним не тільки початковим розробником, але і будь-яким іншим спеціалістом або будь-якою іншою командою. Відсутність прив'язки до основного розробника є вкрай важливим у складних проєктах.
- Висока надійність і супроводжаємость рішень — в цих мовах зазвичай використовується TDD-підхід (TDD — робити речі правильно, на відміну від BDD — робити правильні речі), що робить створювані продукти більш стабільними і супроводжуваними. Функціональність фреймворків також звичайно покрита автоматичними тестами, що робить їх використання дійсно надійним — є впевненість, що нічого не зламається. Для бізнес-проєктів ця складова вкрай важлива — від стабільності роботи додатка часто залежить ефективність роботи бізнесу в цілому.
- Масштабованість, продуктивність і високі навантаження — фреймворки

звичайно «заточені» під розробку додатків, до яких пред'являються високі вимоги щодо доступності: окремі фреймворки відмінно розгортаються і працюють в кластерах серверів або в «хмарах». Для інтернет-сервісів це дуже важливий критерій, оскільки для власної ефективності і для привабливості в очах користувача вони повинні мати високі швидкості розгортання і роботи.

- Компетентні розробники. Частка хороших програмістів на цих мовах набагато вища, ніж на інших мовах. Це є наслідком їх високого рівня, значного поширення та великої кількості корпоративних проєктів та проєктів сховищ даних корпоративних інформаційних систем. Чим компетентніше розробники, тим швидше (дешевше) розробляється проєкт і тим він більш продуманий, масштабується і надійний.

Будемо вважати, що інтегральна оцінка кожного з інструментів може бути обрахована як середнє геометричне якостей кожного продукту (таблиця 2.5):

$$c_d = k_e \cdot c_e + k_q \cdot c_q + k_v \cdot c_v + k_w \cdot c_w + k_r \cdot c_r, \quad (2.5)$$

де: $k_e \cdot c_e$ — ваговий коефіцієнт та коефіцієнт якості щодо захищеності; $k_q \cdot c_q$ — ваговий коефіцієнт та коефіцієнт якості щодо швидкості; $k_v \cdot c_v$ — ваговий коефіцієнт та коефіцієнт якості щодо наочності; $k_w \cdot c_w$ — ваговий коефіцієнт та коефіцієнт якості щодо швидкості обробки запитів; $k_r \cdot c_r$ — ваговий коефіцієнт та коефіцієнт якості щодо валідності.

В результаті здійснення підстановок (2.5), (2.4), (2.3), (2.2), в (2.1) отримаємо формулу розрахунку значення оцінки ефективності конфігурації сховища даних.

Для визначення вагових коефіцієнтів можуть бути взяті дані з таблиць 2.3, 2.4, 2.5 або такі коефіцієнти можуть бути сформовані шляхом проведення експертизи, наприклад, за популярністю кожної з мов програмування або обсягами інвестицій.

2.6. Висновки до розділу 2

В розділі були розглянуті технології організації і доступу до даних у проектах сховищ даних корпоративних інформаційних систем, що визначаються структурою даних у сховищах, розподілом даних між окремими розділами сховища, інструментальним забезпеченням, методами та засобами збереження, транспортування.

Виконано розробку математичної моделі обчислення відповідності технологій аналізу ефективності сховища даних АІС щодо можливостей його реалізації в даній предметній області та можливостей забезпечення необхідних користувацьких якостей.

Розроблено спрощену систему визначення вагових коефіцієнтів оцінки ефективності сховища даних АІС відповідно до властивостей інструментального забезпечення проекту.

РОЗДІЛ 3. ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ ОЦІНКИ ЕФЕКТИВНОСТІ СХОВИЩ ДАНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

3.1. Розробка адаптивної моделі навантаженого сервера активного сховища даних корпоративної інформаційної системи

Враховуючи результати досліджень та з метою перевірки адекватності отриманих рішень розробимо адаптивну модель організації даних для навантаженого сервера даних корпоративної інформаційної системи. В якості основних механізмів забезпечення адаптивності будемо використовувати наступні:

- нормалізація даних в БД (генерація додаткових індексів);
- денормалізація даних в БД (зміна структури БД);
- збереження проміжних результатів (створення тимчасових таблиць).

Для вибору механізмів адаптивного перетворення будемо використовувати результати профілювання СКБД та оригінальні утиліти збору даних, які базуються на часі виконання запитів.

Алгоритми процедур наведені на рис. 3.1 — 3.4.

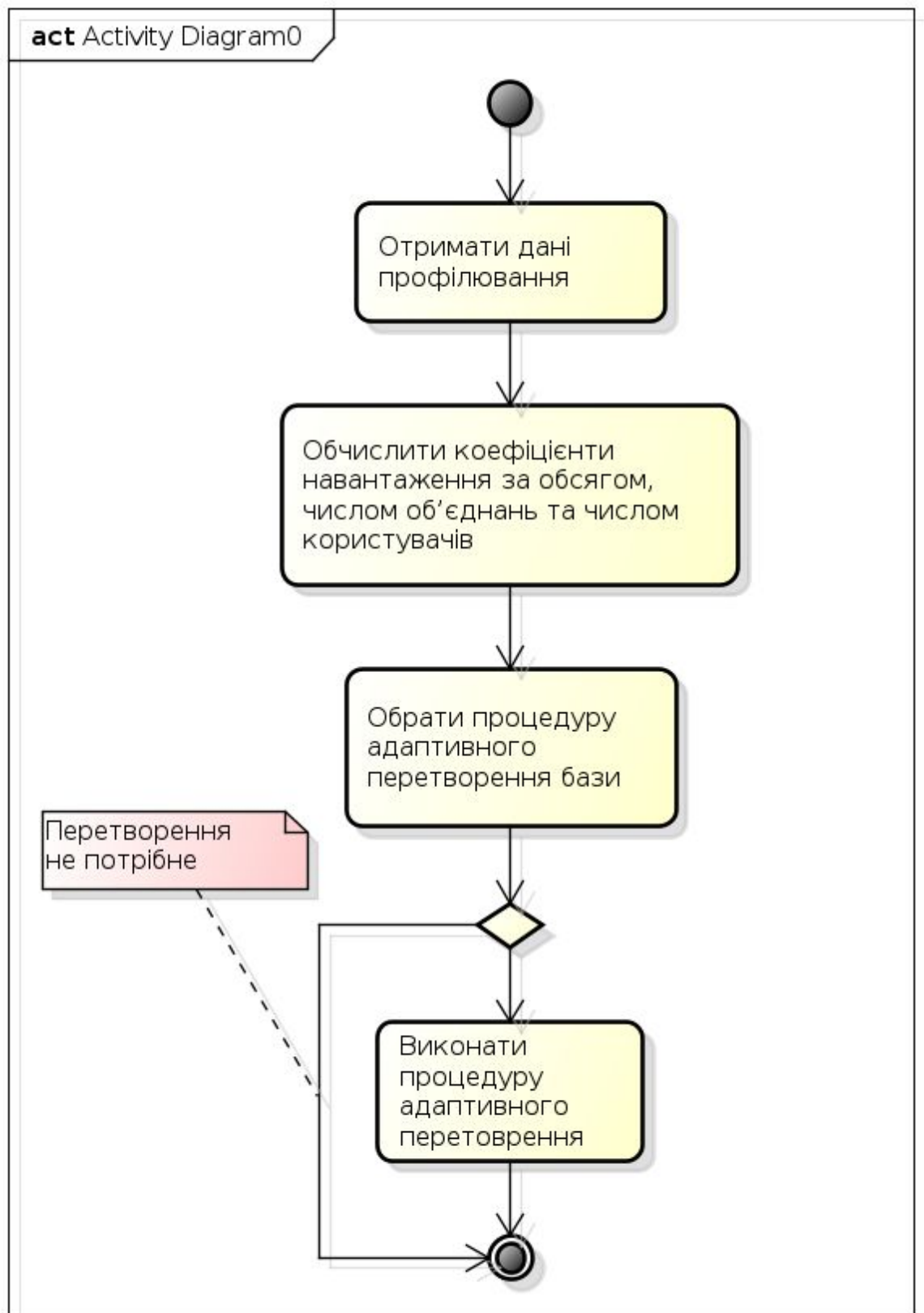
З метою проведення оцінки якості інструментального забезпечення корпоративних інформаційних систем та проєктів сховищ даних корпоративних інформаційних систем був реалізований спрощений алгоритм обчислення значення цільової функції, який виконується методом перебору (Рис. 3.1).

Реалізація виконана мовою програмування Python (Додаток В).

В більш складному випадку (при значному числі інструментальних засобів, обмеженнях у часі тощо) можуть бути застосовані методи вирішення оптимізаційних задач (лінійного програмування, сімплекс методи тощо).

На основі розрахунків, проведених на основі розробленого алгоритму (рис. 3.1) визначено, що у навчальних системах доцільно використовувати хмаро-орієнтовані сховища, зокрема Google Drive, Microsoft OneDrive, Dropbox, які дозволять викладачам зберігати файли поза системою MOODLE, з

можливістю додавати до курсів посилання на ці файли, використовуючи стандартні сховища або вбудовувати вміст цих файлів в деякі ресурси та види діяльності курсу.



powered by Astah

Рис. 3.1. Алгоритм процедури забезпечення адаптивності

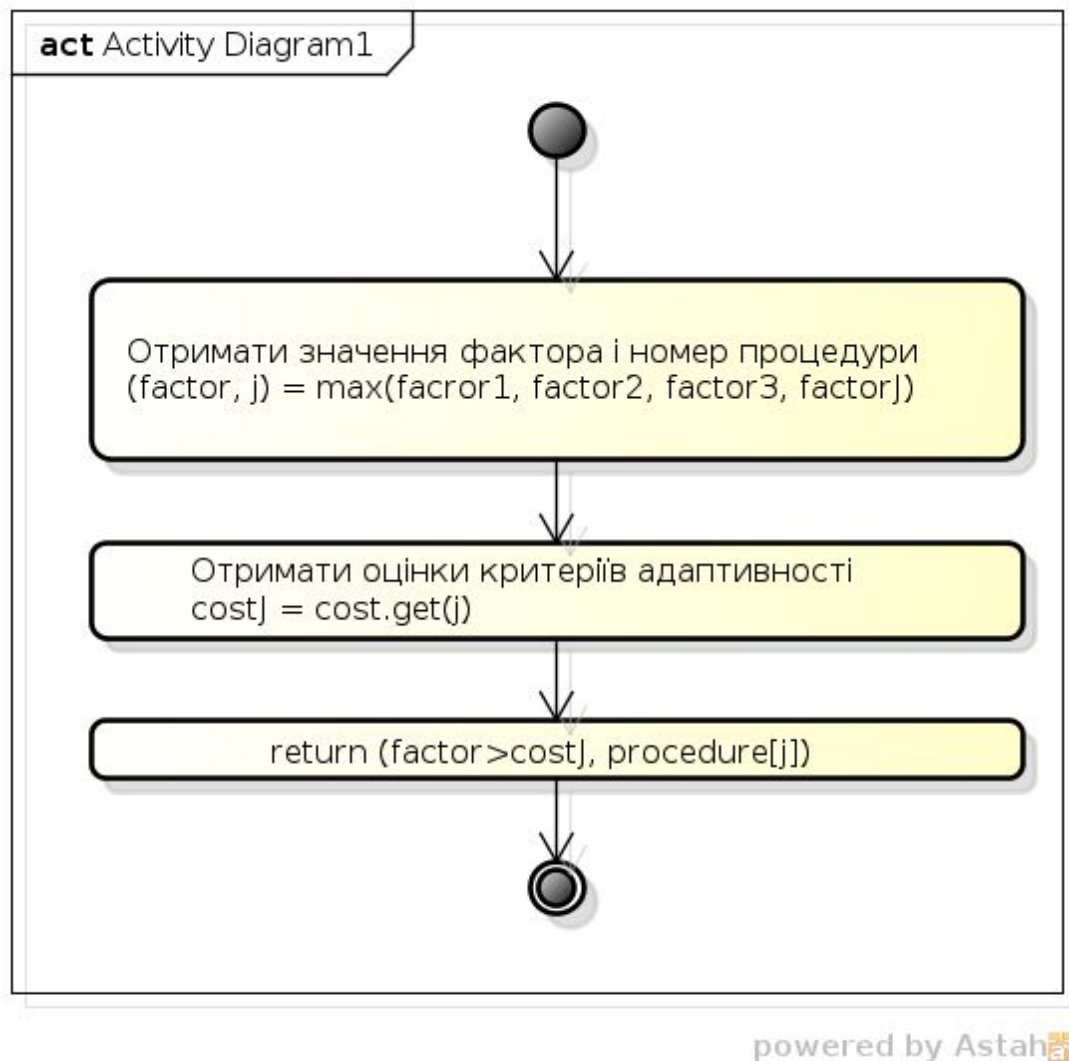
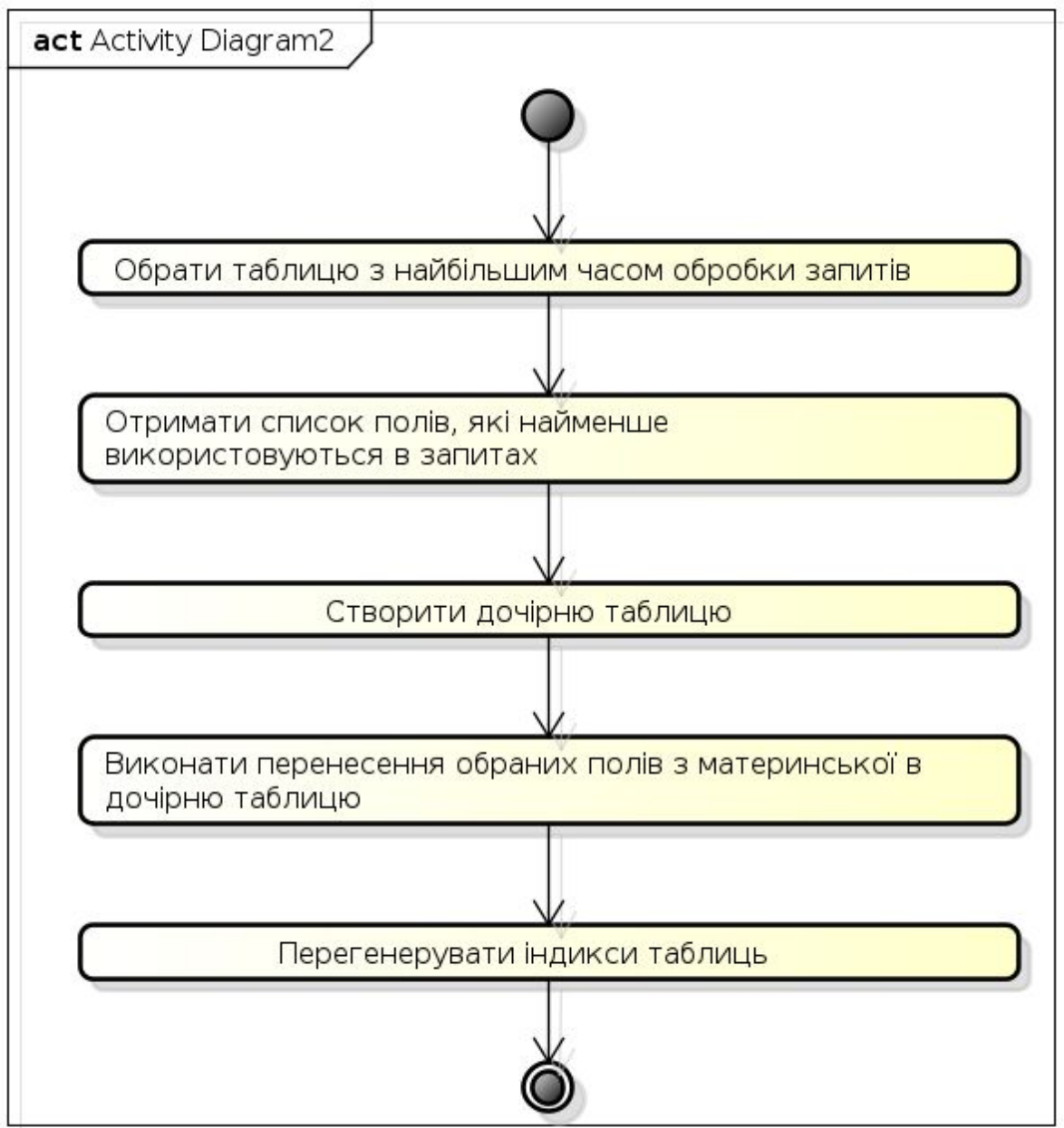


Рис. 3.2. Алгоритм процедури забезпечення адаптивності

Результати досліджень були застосовані в окремих модулях проекту автоматизованої системи управління комунальним господарством.

З метою забезпечення можливостей горизонтальної інтеграції підсистем проектованої автоматизованої системи між собою і вертикальної інтеграції системи в інформаційну систему управління проектами розвитку технічних систем комунального господарства міста, у проекті автоматизованої системи повинні бути передбачені відповідні складності і особливостям вирішуваних завдань механізми та засоби інтеграції підсистем інформаційної системи в єдину систему.



powered by Astah

Рис. 3.3. Алгоритм адаптивної нормалізації за атрибутами



Рис. 3.4. Алгоритм адаптивної кластеризації за серверами

Забезпечення можливостей горизонтальної інтеграції підсистем між собою повинна передбачати такі інструменти реалізації системи, які дозволять досить просто нарощувати функціональність системи шляхом розширення інформаційної бази системи і введення в систему нових модулів. Вертикальна інтеграція системи в інформаційну корпоративну систему повинна передбачати забезпечення таких можливостей доступу до системи з боку зацікавлених осіб і не повинні вимагати істотної модифікації їх інформаційних систем. Це завдання може бути вирішена шляхом надання віддаленого доступу

до ресурсів системи для інших клієнтів системи через локальну мережу або Інтернет із застосуванням поширених і загальнодоступних комунікаційних протоколів для передачі даних і управління між підсистемами корпоративних інформаційних систем.

Локальний доступ може забезпечуватися через автоматизовані робочі місця (АРМ), встановлені на клієнтських робочих місцях персоналу підприємства. Для цього на клієнтські робочі місця повинно бути встановлено відповідне функціям клієнтів клієнтське програмне забезпечення, а клієнтам повинні бути надані права і повноваження по доступу до інформаційних ресурсів сервера інформаційної системи відповідно до функціональних обов'язків кожного клієнта системи.

Враховуючи, що в кожній інформаційній системі накопичується інформація про параметри інформаційної системи, яка може використовуватися як в поточній, так і в інших системах, необхідно розглянути механізми доступу до інформаційних ресурсів системи з боку інших систем. Для цього може використовуватися глобальний (віддалений) доступ через мережу Інтернет, у тому числі – із застосуванням протоколів TCP / IP і HTTP.

У результаті забезпечення зазначених механізмів доступу, доступ до інформаційних ресурсів системи може бути наданий для наступних видів користувачів системи:

- авторизованих користувачів – менеджерів підприємства, команд управління проектами (надання прав і повноважень доступу до інформаційних ресурсів системи повинне буде забезпечуватися як для режиму локального доступу);
- авторизованих користувачів – співробітників розрахункових центрів комунальних служб водопостачання (авторизація виконується на основі відомої користувачам приватної облікової інформації: адреса, номер розрахункової книжки тощо);
- інших користувачів – інформація надається без авторизації (оголошення,

таблиці-графіки ремонтних робіт, параметри води і т.п.).

На основі вищенаведеного був розроблений програмний продукт як складовая наявної АІС для забезпечення міграції структури даних сховища даних АІС. Фрагмент коду одного з класів мовою програмування Python 3 наведений на лістингу 3.1.

```
class MetaModelCluster(object):
    """провайдер доступу до даних таблиці або групи таблиць БД через їх моделі"""
    # .....

class MetaModelResult(dict):
    """провайдер верифікації таблиць або груп таблиць БД за їх моделями"""
    def __init__(self, Db, obj):
        self._Db = Db
        self._data = obj.copy()
        self._counters = {}

    def select(self, query):
        result, tim = Db.exec_select(query)
        tablename = set([ table._name for table in query._froms ])
        for tablename in tablename:
            self._counters[tablename]['count'] += 1
            self._counters[tablename]['tim'] += tim
        return result

    def split(self, otherDb=None, cost=1.0):
        """створює об'єкт відокремленої таблиці для зменшення обсягу
        невикористаних даних"""
        mintablename = self._getMintable(self._counters)
        avgTableUse = self._getAvgTableUse(mintablename, self._counters)
        If avgTableUse < cost:
            self._tableMove(mintablename, self._Db, otherDb)
            self._tableLink(mintablename, self._Db, otherDb)
            return True
        return False
```

Лістинг 3.1. Код модуля кластеризації даних (фрагмент)

3.2. Результати тестування активного сховища даних для різних “engines” РСКБД MySQL

На основі результатів дослідження розробимо рекомендації з вибору

конфігурації системи (рис. 3.2):

- Визначити можливість використання монітору транзакцій.
- Визначити можливість використовувати систему в конфігурації клієнт-сервер.
- Визначити число одночасно активних користувачів.
- Визначити стратегії індексації даних.
- Визначити чистий розмір бази даних.
- Визначити ємність та швидкість виконання операцій читання-запису для дискової пам'яті необроблених даних, індексів, тимчасових табличних просторів.
- Узгодити передбачувану стратегія резервного копіювання.

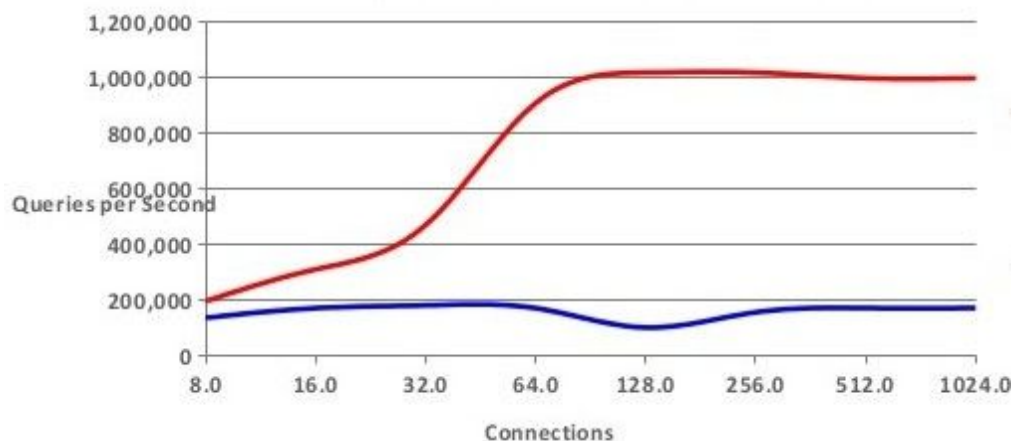


Рис. 3.2. Тести навантаження сервера (внизу з фіксованою структурою, зверху з вибором структури сховища даних за результатами аналізу)

3.3. Висновки до розділу 3

В розділі було запропоновано спрощений алгоритм пошуку рішення на основі цільової функції оцінки якості, розглянуті можливості її застосування щодо проєктів з декількох предметних областей.

Розроблені пропозиції щодо використання інструментальних засобів в різних корпоративних інформаційних системах та проєкт сховища даних

корпоративної інформаційної системах, а також забезпечення адаптивності структури БД.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

В роботі в цілому вирішені задачі дослідження, що дозволило досягти основної мети дослідження: підвищення ефективності процесів аналізу та оцінки ефективності корпоративного сховища даних для прийняття рішень щодо забезпечення актуальності даних та реорганізації структури сховища даних в клієнто-орієнтованих компаніях на основі науково обґрунтованих моделей та механізмів.

Запропоновані рішення можуть бути застосовані при вирішенні задач підвищення ефективності сховищ даних в корпоративних автоматизованих інформаційних системах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гольдінер Д. І. Розробка архітектури програмного забезпечення для моделювання систем масового обслуговування під імплементацію мовою програмування Go. Вісник Національного технічного університету «ХПІ». 2024. № 1 (11). С. 85–90.
2. Коваленко В.В., Гуменюк В.О. Аналіз найактуальніших серверних систем управління базами даних. В.В. Коваленко, В.О. Гуменюк, — Харків. Національний авіаційний університе 2007.
3. Мак-Федрис П. Использование JavaScript. Специальное издание. — Киев. Диалектика, 2002.
4. Склярєнко О.В., Савченко Я.О., Литвиненко Л.О., Сушинський О.Є. Архітектурні підходи до розробки масштабованих веб-застосунків. Cybersecurity Education Science Technique. 2024. Vol. 4, No. 24. P. 341–350.
5. Abbot M. L., Fisher M. T. The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise. Addison-Wesley, 2015. 624 p.
6. Albada O. Architecture in Practice: Challenges and Opportunities in the Age of Digital Transformation. Department of Software Engineering, Palestine Technical University-Kadoorie (PTUK), Tulkarm, Palestine. 2023. November. P. 1–1
7. Colley D., Stanier C., Identifying new directions in database performance tuning. Procedia Computer Science, vol. 121, p. 260– 265, 2017.
8. El Murabet A., Abtoy A. Methodologies of the Validation of Software Architectures. Journal of Computing Theories and Applications. 2023. Vol. 1, No. 2. P. 29–36.
9. Erl T. Service-Oriented Architecture: Analysis and Design for Services and Microservices. Boston: Prentice Hall, 2016. 864 p.
10. Factor Ph. Тимчасові таблиці. URL: <https://www.red-gate.com>

/simple-talk/sql/t-sql programming/temporary-tables-in-sql-server/ (Last accessed: 08.10.2024)

11. Gorton I. Foundations of Scalable Systems. New York: O'Reilly Media, 2022. 352 p.
12. Introduction to High Load: What Is It? Geniusee. URL: <https://geniusee.com/single-blog/introduction-to-high-load-what-is-it> (Last accessed: 08.11.2024).
13. Optimization Model. IBM. URL: <https://www.ibm.com/topics/optimization-model> (Last accessed: 20.11.2024).
14. Saaty T. L. The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. New York: McGraw-Hill, 1980. 287 p.
15. Software Architecture Resources. IEEE Computer Society. URL: <https://www.computer.org/resources/software-architecture> (Last accessed: 15.11.2024).
16. VanDyk John K. Pro Drupal 7 Development: Third Edition / Todd Tomlinson . John K. VanDyk — Apress, 2010.
17. Woods E. Software Architecture in a Changing World. IEEE Software. 2016. Vol. 33, No. 6. P. 94–97.

ДОДАТОК А.

**ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО
ЗАБЕЗПЕЧЕННЯ ПІДСИСТЕМИ АНАЛІЗУ ЕФЕКТИВНОСТІ
СХОВИЩА ДАНИХ**

Вступ

Назва розробки “Підсистема аналізу ефективності сховища даних корпоративної автоматизованої інформаційної системи”.

Призначенням розробки – автоматизація бізнес-процесів аналізу ефективності засобів збереження і доступу до даних корпоративної автоматизованої інформаційної системи.

Вимоги до проекту

Вимоги до функціональних характеристик програмного забезпечення:

- Автоматичний розрахунок коефіцієнтів навантаження на окремі сервери баз даних за таблицями, атрибутами таблиць і бізнес-процесами.
- Формування і надання звітів про розподіл навантаження між окремими серверами баз даних за таблицями, атрибутами таблиць і бізнес-процесами.
- Виконання ручного і автоматичного розподілення даних між серверами СКБД за результатами аналізу навантаження на сервери баз даних.
- Виконання процесів ручного злиття або міграції даних з серверів СКБД на вказаний сервер.

Вимоги до надійності і якості

Надійність і якість функціонування проекту повинні забезпечуватися:

- оптимізованою структурою проекту;
- системою виводу повідомлень про можливі загрози;
- наявністю дублікатів проекту і зберіганням програмного забезпечення на

магнітних носіях.

Умови експлуатації

Програмне забезпечення розрахована на користувача з кваліфікацією роботи з СКБД на рівні адміністратора баз даних.

Вимоги до складу і параметрів апаратного забезпечення підсистеми аналізу ефективності

Проект розрахований на роботу в локальній чи Інтернет мережі. Мінімальні вимоги функціонування проекту:

- Процесор із частотою 3,2 ГГц
- Оперативна пам'ять 4 Гб
- Відео пам'ять 32 МБ
- Мережева карта не нижче 1 Гб
- Жорсткий диск 2 ТБ

Вимоги до інформаційної та програмної сумісності

Проект повинен задовольняти вимогам використовуваної операційної системи, бути сумісними із системними програмними засобами і СКБД, мати можливість переносу на інші ЕОМ аналогічного класу.

Вимоги до випробування

Випробування має виконуватись на запропонованому замовником контрольному прикладі. Контрольний приклад повинен включати тестові завдання на фрагментацію і дефрагментацію бази даних і забезпечувати перевірку правильності й працездатності функціонального наповнення проекту і його програмної реалізації, забезпечувати перевірку працездатності програм у різних ситуаціях.

Контрольний приклад має охопити всі варіанти рішення, передбачені

функціональним описом і алгоритмами, відбивати всі етапи рішення, демонструвати можливості проекту.

Вимоги до програмної документації

Вимоги до складу, утримування й оформлення програмної документації визначаються стандартами ЕСПД. Склад програмної документації:

- Опис програми.
- Опис застосування.
- Керівництво системного програміста.
- Функціональний опис.
- Зміст програми на носії даних.

Стадії та етапи розробки

Стадії та етапи розробки підсистеми підсистеми аналізу ефективності сховища даних, зміст робіт і терміни виконання наведені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадії	Етапи роботи	Початок робіт	Закінчення робіт
1. Аналіз	Виконання аналізу і постановка задачі	12.09.2024	26.09.2024
2. Дослідження та розробка моделей	Дослідження методів забезпечення комунікацій та ефективності в активних сховищах даних	27.09.2024	12.10.2024
	Затвердження концепції та технічного завдання	13.10.2024	17.10.2024

	Розробка проектних рішень по видам забезпечення системи	18.10.2024	31.10.2024
	Затвердження проектних рішень	01.11.2024	06.11.2024
3. Робочий проект	Розробка програмного забезпечення підсистеми аналізу ефективності сховища даних, розгортання підсистеми	07.11.2024	30.11.2024
	Розробка документації по системі	01.12.2024	08.12.2024
	Випробування програми	09.12.2024	10.12.2024

ДОДАТОК Б.

ПРИКЛАД РЕАЛІЗАЦІЇ МОДЕЛЕЙ ДАНИХ З МЕХАНІЗМАМИ ПІДСИСТЕМИ АНАЛІЗУ ЕФЕКТИВНОСТІ СХОВИЩА ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ (ФРАГМЕНТ)

```
class ProxyBase(Proxy):
    def __init__(self, adopt=True):
        Proxy.__init__(self)
        self._adopt = adopt

    def _has_adopt(self, kb, cub, kp, cup):
        if not self._adopt:
            return 0.0
        return kb * cub + kp * cup

    def adopt(self, modelProxynome, kb, cub, kp, cup, limit=0.1):
        adopt_weight = self._has_adopt(kb, cub, kp, cup)
        if adopt_weight < limit:
            return
        modelProxyCls = getProxyByProxynome(modelProxynome)
        tmpModel = createTmpModel(modelProxynome)
        adoptModel(modelProxyCls, tmpModel, adopt_weight)
        restoreData(modelProxyCls, tmpModel)

class PhysPlacementProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, \
            Cluster((dsab.Placement & dsab.Abonent ^ "physabonent") & \
                dsab.PhysAbonent & dsab.Apartment & dsab.Building & \
                dsab.Street & dsab.Location, \
                None, lambda p, a, pa, apart, b, s, l: a.typ=="phys"))

class JurPlacementProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, \
            Cluster((dsab.Placement & dsab.Abonent ^ "jurabonent") & \
```

```

        dsab.JurAbonent & (dsab.JurAbonentApartment & \
dsab.Apartment) & dsab.Building & dsab.Street & \
        dsab.Location, None, \
        lambda p, a, ja, jabonapart, apart, b, s, l: a.typ=="jur"))

def save(self, modelname, vals):
    apartmentids = vals.pop('JurAbonentApartment.apartment')
    items = self._cluster.distribute_values(vals)[modelname]
    placement = self._models[modelname](**items).save()
    apartments = self.save_jurabonentapartments(placement.id, \
apartmentids)
    if placement.is_saved():
        self.facade.sendNotification('%sisUpdate' % modelname)
    else:
        self.facade.sendNotification('%sUpdateError' % modelname)
    return placement.id

```

```

def save_jurabonentapartments(self, placementid, apartmentids):
    apartments = []
    dsab.JurAbonentApartment.remove(placement=placementid)
    for apartmentid in apartmentids:
        apartments.append( \
            dsab.JurAbonentApartment(apartment=apartmentid, \
placement=placementid).save())
    return apartments

```

```

def delete(self, modelname, id):
    dsab.JurAbonentApartment.remove(placement=id)
    super(JurPlacementProxy, self).delete(modelname, id)

```

```

def getVO(self, fields, filt=None, direction='top', order=None):
    return ProxyBase.getVO(self, fields, filt=filt, \
        direction=direction, order=order, distinct=True)

```

```

class PhysAbonentProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, \

```

```
Cluster(dsab.Abonent & dsab.PhysAbonent, None))
```

```
def save(self, modelname, vals):
    items = self._cluster.distribute_values(vals)[modelname]
    abon = self._models[modelname](**items).save()
    phys = self.save_phisabonent(abon.id, vals)
    if abon.is_saved() and phys.is_saved():
        self.facade.sendNotification('%sIsUpdate' % modelname)
    else:
        self.facade.sendNotification('%sUpdateError' % modelname)
    return abon.id
```

```
def save_phisabonent(self, abonid, vals):
    vals['PhysAbonent.abonent'] = abonid
    items = self._cluster.distribute_values(vals)["PhysAbonent"]
    return self._models["PhysAbonent"](**items).save()
```

```
def delete(self, modelname, id):
    """удаляет объект модели @modelname по @id"""
    o = self._models[modelname].get(id)
    try:
        o._get_reference().delete()
        o.delete()
        self.facade.sendNotification('%sIsDelete' % modelname)
    except:
        self.facade.sendNotification('%sDeleteError' % modelname, \
            o.require_related())
```

```
class JurAbonentProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, Cluster(dsab.Abonent & dsab.JurAbonent, \
            None))
```

```
def save(self, modelname, vals):
    items = self._cluster.distribute_values(vals)[modelname]
    abon = self._models[modelname](**items).save()
    jur = self.save_jurabonent(abon.id, vals)
```

```
if abon.is_saved() and jur.is_saved():
    self.facade.sendNotification('%sIsUpdate' % modelName)
else:
    self.facade.sendNotification('%sUpdateError' % modelName)
return abon.id
```

```
def save_jurabonent(self, abonid, vals):
    vals['JurAbonent.abonent'] = abonid
    items = self._cluster.distribute_values(vals)["JurAbonent"]
    return self._models["JurAbonent"](**items).save()
```

```
def delete(self, modelName, id):
    """удаляет объект модели @modelName по @id"""
    o = self._models[modelName].get(id)
    try:
        o._get_reference().delete()
        o.delete()
        self.facade.sendNotification('%sIsDelete' % modelName)
    except:
        self.facade.sendNotification('%sDeleteError' % modelName, \
            o.require_related())
```